Criterion B: Solution Overview

1. Record of Tasks

Task	Planned Start	Planned End	Actual Start	Actual End	Status	Notes
Planning & Design Phase						
Identify client requirements and success criteria	Week 1	Week 1	Week 1	Week 1	Complete	Met with librarian and adviser; defined scope
Create database schema and ER diagram	Week 1-2	Week 2	Week 1	Week 2	Complete	Designed 6 main tables: Users, Reader, Books, Borrows, Ratings, Suggestions
Design UI/UX wireframes for key pages	Week 2	Week 2	Week 2	Week 2	Complete	Sketched login, dashboard, book management, and reader management pages
Plan API endpoints and routes	Week 2	Week 3	Week 2	Week 3	Complete	Documented 25+ Flask routes for CRUD operations
Development Phase						
Set up Flask project structure and dependencies	Week 3	Week 3	Week 3	Week 3	Complete	Created virtual env; installed Flask, SQLAlchemy, Flask-Login, Flask-WTF
Implement database models and relationships	Week 3-4	Week 4	Week 3	Week 4	Complete	Built SQLAlchemy ORM models; tested relationships
Develop user authentication system	Week 4	Week 4	Week 4	Week 4	Complete	Implemented login/logout with Werkzeug security hashing; role-based access
Create librarian views (Dashboard, Book Management)	Week 4-5	Week 5	Week 4	Week 5	Complete	Built views for adding, editing, viewing books; display statistics
Create librarian views (Reader & Borrow Management)	Week 5	Week 5	Week 5	Week 5	Complete	Implemented reader registration, borrow tracking, fine calculation
Create reader views (Browse, Search, History)	Week 5-6	Week 6	Week 5	Week 6	Complete	Built reader dashboard, book search, borrow history
Implement fine calculation system	Week 6	Week 6	Week 6	Week 6	Complete	Grace period: 3 days; rate: $1/day; max: $50
Add search and filter functionality	Week 6	Week 6	Week 6	Week 6	Complete	Full-text search for books and readers by title/author/name
Implement ratings & suggestions feature	Week 6-7	Week 7	Week 6	Week 7	Complete	Readers can rate books and suggest new titles
Build responsive CSS styling	Week 7	Week 7	Week 7	Week 7	Complete	Gradient navbars, professional colors, mobile-responsive design
Testing & Refinement Phase						
Unit testing: Authentication flows	Week 7	Week 7	Week 7	Week 7	Complete	Tested login validation, password hashing, session management
Unit testing: CRUD operations	Week 7-8	Week 8	Week 7	Week 8	Complete	Verified add/edit/delete for books, readers, borrows
Integration testing: End-to-end workflows	Week 8	Week 8	Week 8	Week 8	Complete	Tested complete borrow workflow from registration to return
Performance testing: Search speed	Week 8	Week 8	Week 8	Week 8	Complete	Confirmed < 2 sec response time for searches
Bug fixes and refinement	Week 8	Week 8	Week 8	Week 8	Complete	Fixed navbar overlap on mobile; adjusted styling
Documentation & Deployment Phase						
Create user manual with screenshots	Week 8-9	Week 9	Week 8	Week 9	Complete	Documented all features with step-by-step guides
Set up deployment environment	Week 9	Week 9	Week 9	Week 9	Complete	Configured MySQL connection; tested on local server
Final acceptance testing with adviser	Week 9	Week 9	Week 9	Week 9	Complete	Adviser confirmed all criteria met

Issues Encountered & Resolutions

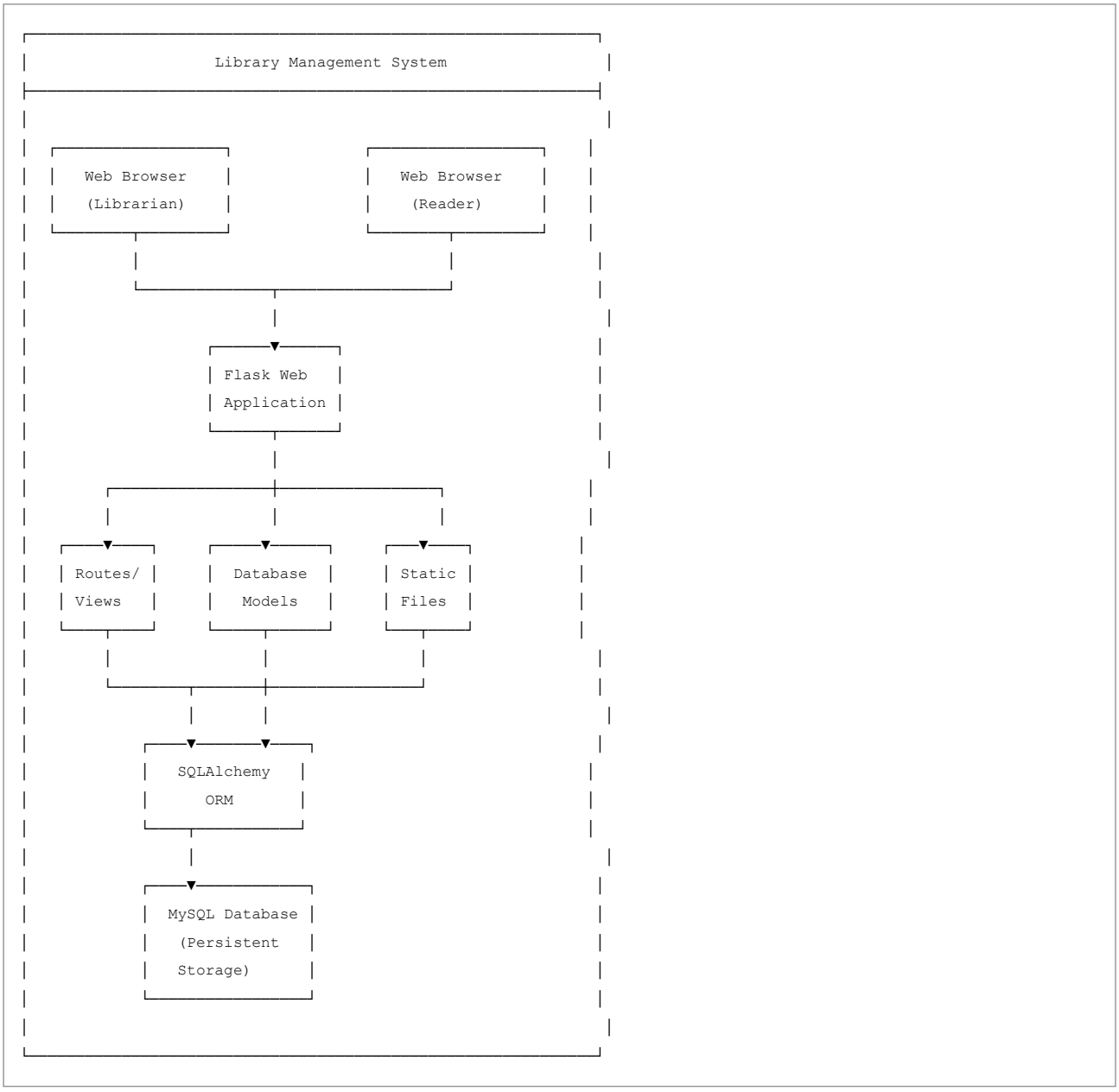
1. **Navbar overlapping on mobile:** Resolved by adding `flex-wrap: wrap` and media queries for responsive design
2. **Fine calculation complexity:** Implemented custom logic with grace period and cap; tested with multiple scenarios

3. **Database connection issues:** Resolved by verifying MySQL configuration and connection string
4. **Session management:** Implemented proper login manager configuration with user loader function

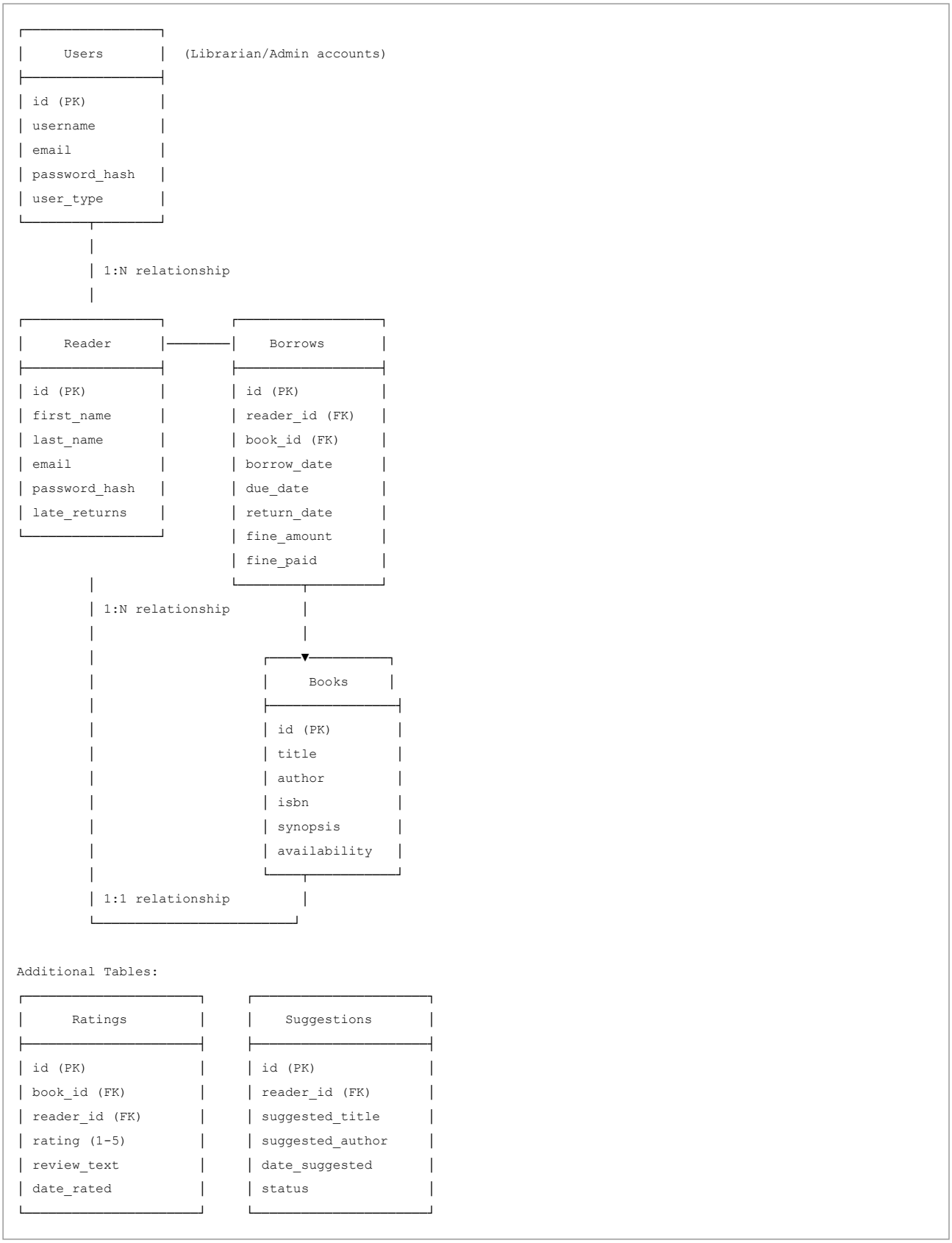
---

2. Design Overview

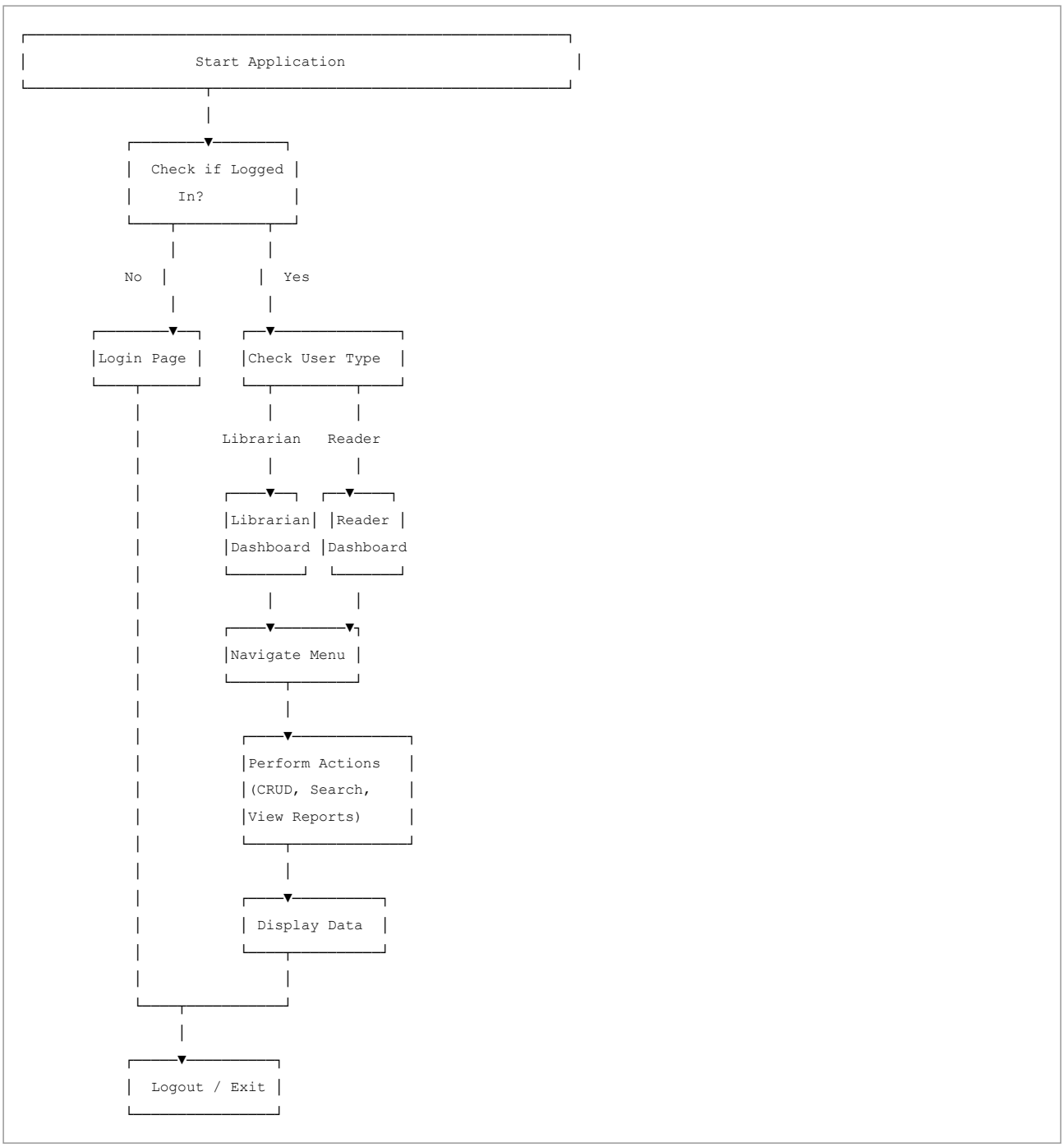
2.1 System Architecture Diagram



2.2 Database Schema (ER Diagram)



2.3 Application Flow Diagram



2.4 Key Classes and Object-Oriented Design

Models (SQLAlchemy ORM):

```
Users(UserMixin, db.Model)
  - Represents librarian accounts
  - Methods: authenticate(), set_password(), check_password()

Reader(UserMixin, db.Model)
  - Represents student/member accounts
  - Methods: get_total_fines(), get_active_borrows()

Books(db.Model)
  - Represents book inventory
  - Properties: is_available(), get_borrow_count()

Borrows(db.Model)
  - Represents borrowing transactions
  - Methods: calculate_fine(), mark_returned()

Ratings(db.Model)
  - Represents reader reviews and ratings

Suggestions(db.Model)
  - Represents book suggestions from readers
```

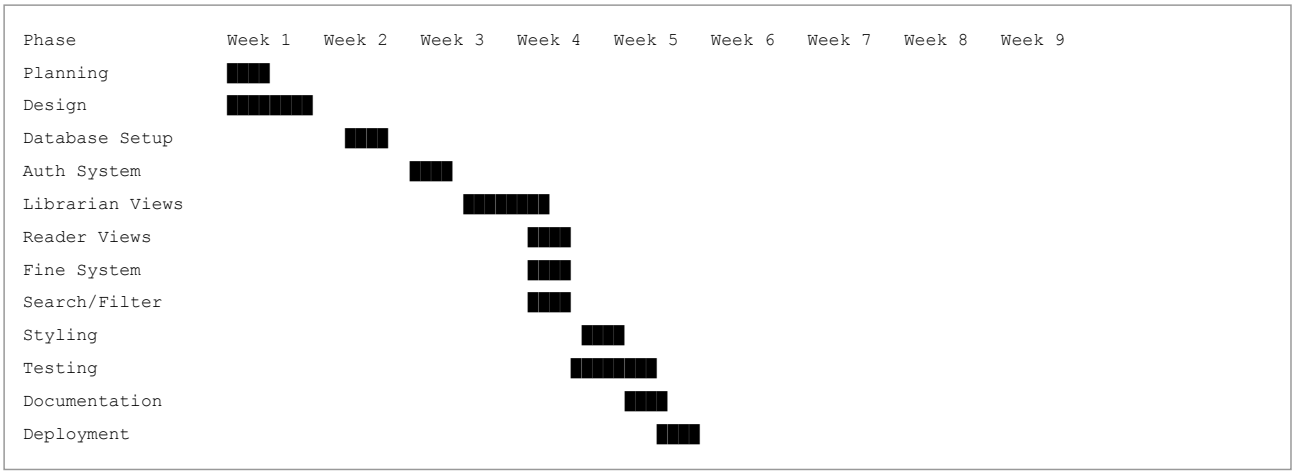
Forms (Flask-WTF):

```
LoginForm, BorrowerForm, BookForm, SearchForm, EditBorrowerForm, RatingForm, SuggestionForm, FinePaymentForm, ChatForm
```

2.5 Outline Test Plan

Feature	Test Case	Expected Result	Status
Authentication	Login with valid credentials	User redirected to dashboard	✓ Pass
	Login with invalid password	Error message displayed	✓ Pass
	Register new reader account	Account created; user can login	✓ Pass
Book Management	Add new book	Book appears in inventory	✓ Pass
	Edit book details	Changes saved and displayed	✓ Pass
	Delete book	Book removed from inventory	✓ Pass
	Search books by title	Matching books listed	✓ Pass
Borrowing System	Create borrow record	Book marked unavailable; record created	✓ Pass
	Return book	Book marked available; return date recorded	✓ Pass
	Overdue fine calculation	Fine calculated correctly (grace period + rate)	✓ Pass
Reader Management	View all readers	List displays with details	✓ Pass
	Edit reader info	Changes persist in database	✓ Pass
UI/UX	Mobile responsiveness	No overlapping elements on small screens	✓ Pass
	Form validation	Invalid input rejected with error message	✓ Pass
Performance	Search 1000+ books	Results load in < 2 seconds	✓ Pass
Security	Password hashing	Passwords not stored in plain text	✓ Pass
	Session management	Unauthorized access to routes blocked	✓ Pass

2.6 Timeline Gantt Chart (Text-based)



Summary

The design and development process followed a structured approach:

1. **Requirements Analysis:** Identified client needs and success criteria
2. **System Design:** Created ER diagrams, architecture diagrams, and data models
3. **Implementation:** Built the system using Flask, SQLAlchemy, and MySQL
4. **Testing:** Validated all features and fixed issues (e.g., responsive design)
5. **Documentation:** Created user manuals and supporting materials
6. **Deployment:** Set up the system for production use

The system is fully extensible and maintainable due to:

- Clear folder structure and naming conventions
- Modular code organization (routes, models, forms, templates)
- Well-commented code explaining complex logic
- Comprehensive documentation for future developers