Library Data Warehouse & Reporting System

Project Layout & Design Concept (Client → Developer)

January 29, 2026

1. Executive Summary

This document provides a **project layout idea** and **design concept** for a Library Data Warehouse solution. It is written as a **client-to-developer** brief: what the client wants, translated into implementable components for the developer.

2. Business Context & Problem

The existing library system is an **operational database (OLTP)** intended for transactions (loans, returns, reservations). It is not designed for analytical reporting. The client requires a **warehouse (OLAP)** and an application that can answer decision-maker questions reliably and quickly.

3. Stakeholders

Stakeholder	Type	What they need
Chief Librarian	Owner	Service usage insights, policy and resourcing decisions.
Finance Director	Owner	Fine totals, trends, and budgeting indicators.
Department Heads	Consumer	Usage by faculty/course, student engagement signals.
Analyst/Report user	Consumer	Filterable, exportable reports and dashboards.
System Administrator	Technical	User management, roles, backups, monitoring.

4. Goals (In Scope)

- Operational database schema + representative sample data.
- Warehouse database using a dimensional model (facts + dimensions).
- ETL process to load operational data into the warehouse.
- Secure GUI application providing interactive reports.
- Evidence of indexes, prepared statements, and performance/security considerations.

5. Out of Scope (Unless explicitly required)

- Real-time streaming / live dashboards.
- Integration with external identity providers (SSO).

- ML-based recommendations.

6. Functional Requirements

6.1 Login & Roles

- Users must authenticate.
- Authorisation is role-based (RBAC), e.g., `ADMIN`, `ANALYST`, `VIEWER`.
- Admin users can create/disable accounts and assign roles.

6.2 Reporting

- Provide interactive reports aligned to decision-maker questions.
- Support filtering by time (e.g., last 1/3/6 months) and key dimensions (faculty, course, category).
- Support comparison across dimensions (tables and/or charts).
- Export results (CSV and/or PDF depending on toolchain).

6.3 ETL

- Extract from operational DB, transform to warehouse format, load into dimensional model.
- Load dimensions first, then fact tables.
- Keep a minimal ETL log (run timestamp, inserted/updated counts, errors).

7. Non-Functional Requirements

- **Security:** hashed passwords; least-privilege DB accounts; prepared statements.
- **Performance:** warehouse queries should be fast for sample dataset; appropriate indexing.
- **Reliability:** documented backup/restore/recovery procedure.
- **Maintainability:** clean folder structure; repeatable scripts; configuration separated.

8. Proposed System Architecture

1. **Operational Database (OLTP)**: normalised schema for transactions.
2. **Warehouse Database (OLAP)**: dimensional schema for analytics.
3. **ETL Layer**: repeatable load scripts/code.
4. **GUI Application**: authentication + reporting UI + data access layer.

9. Data Model Concept

9.1 Operational (example entities)

- Student, Course, Faculty
- Book, Author, Category, Copy

- Loan, Reservation, Fine
- StaffUser (or AppUser)

9.2 Warehouse (dimensional model)

Recommended grain: one row per **loan transaction**.

Dimensions (examples)

- **dimDate**: day, month, quarter, year.
- **dimStudent**: student attributes + course + faculty (or snowflaked).
- **dimBook**: title, author, category, publisher.
- **dimLocation**: branch, section.

Fact table (example)

- **factLoan**: measures such as `loanCount=1`, `daysBorrowed`, `overdueFlag`, `fineAmount`.

10. Example Decision-Maker Questions (10)

1. Total loans by month (trend) for the last 6 months.
2. Loans by faculty and month (comparison).
3. Loans by course and month.
4. Top 10 categories by loans (overall and by faculty).
5. Overdue rate by course.
6. Average borrowing duration by category.
7. Fine total by month and faculty.
8. Distinct borrowers per month (new vs returning).
9. Most borrowed authors by faculty.
10. Branch/location demand by month.

11. GUI Concept (Screens & Navigation)

1. **Login** → role loaded.
2. **Dashboard** → quick KPIs + report shortcuts.
3. **Reports** → select report, set filters (date range, faculty, course, category), run.
4. **Report Results** → table + optional chart + export.
5. **Admin** (ADMIN only) → manage users/roles + view ETL/audit logs.

12. Security Concept (Implementation Notes)

- Password storage: salted hash (e.g., bcrypt/Argon2 depending on stack).
- Application uses parameterised queries / prepared statements only.
- Separate DB credentials for: application runtime vs ETL jobs.
- Minimal audit logging: user, action, timestamp.

13. Performance Concept (Warehouse)

- Index foreign keys on fact tables (dateKey, studentKey, bookKey, locationKey).
- Consider partitioning by date for large fact tables (optional).
- Consider summary tables/materialised views for heavy aggregates (optional).

14. Suggested Project Folder Layout

```
/CT6049-Assignment-002/
  /docs/
    assignment-brief.pdf
    project-concept.pdf
  /db/
    /operational/
      schema.sql
      sample-data.sql
    /warehouse/
      schema.sql
      indexes.sql
    /etl/
      etl.sql (or etl.java)
      etl-config.properties
      etl-log.sql
  /app/
    /src/
      /ui/
      /service/
      /dao/
      /security/
      /model/
      Main.java
    /resources/
      application.properties
  /tests/
  README.md
```

15. Deliverables & Acceptance Criteria

- Scripts build both databases from scratch.
- Sample data loads into operational DB.

- ETL loads dimensions and facts; logs runs.
- GUI enforces RBAC and runs reports correctly.
- Documentation matches implementation (setup, credentials, backups, testing).