Assessment Brief

CA1 – Database Design Project

Module Name	Lecturer Name
Advanced Databases	Michael McAndrew

Title of Brief
CA1 – Database Design Project

Percentage of Overall Grade	65%
Date handed out	10th November 2025
Due date	12th December 2025
Individual or Group	Group

Problem to Solve

Letterboxd is a social network for films. Users track films they've watched, maintain film lists, write reviews, follow friends, and discover new films through social recommendations. You are hired as an early engineer for Letterboxd, and **your task is to build the database system for them with MySQL.**

You are **provided with an existing movie database** containing data from The Movie Database (TMDB) dating back to 1970. Your task is **to extend this database** by designing and implementing the user-facing features of Letterboxd: user management, reviews, ratings, social interactions, and lists.

Specification

Part 1: Database Design [20%]

To begin, you must design a database to support the major features of Letterboxd. **Draw an Entity-Relationship Diagram (ERD)** of your database system (including tables from the existing database) and write the table creation queries to the `schema.sql` file provided in the assignment template. Be sure to define schema for tables **precisely**, to make the

database robust against **data integrity issues and well-normalised**. The features that your database design should support are:

- **Users:** usernames, emails, passwords and profile information. Consider what additional metadata might be useful (join date, bio, if they're a premium or regular user etc)
- **Film reviews and logs:** Users can log a movie watch with a star rating, a written review, whether they liked the movie, and a list of tags. They can create a watch log of the same movie as many times as they like. Other users can like their reviews and comment on them.
- **Social Features:** Users can follow each other. Following works one-way, similar to a social network like Instagram
- **Watchlist and Lists:** Each user has their own personal watchlist of movies they want to watch in future. Users can also create custom lists of movies. They can be public or private, have a list of tags associated with them and contain an **ordered list** of movies.

Part 2: Querying the Database [25%]

Users will use the Letterboxd client applications on Android, iOS, and on the Web. The application code will query the database that you built to read, insert, update, and delete data so that users can interact with the platform quickly. The features that your queries should support are:

- **Users:** write queries to create a new account, get user profile information by user ID (including their follower and following count), and update a user profile
- **Reviews and logs:** create a new movie log entry with an optional rating, like, and written review. If a user logs a movie and it is in their watchlist, make sure it is automatically removed from their watchlist
- **Search Functionality:** Search is one of the most important features in Letterboxd. Implement queries to search for a movie, actor, or crew member by name. Consider partial matches, and whether search queries should be case-sensitive or not
- **Social Features:** write queries for one user to follow another, get movie logging activity from users that a given user follows
- **List Management:** write queries to create a new list, a query to add a movie to a user list, and to get a user's watchlist with movie details
- **Movie Statistics:** get the average rating for a given film, and summary statistics about a film: the total number of watches, number of likes, number of lists that a film appears in

Implement these queries using **an appropriate mix of queries, stored procedures, triggers, and views**. Include a brief comment above each explaining your choice.

Part 3: Query Optimisation [10%]

It is important as the database grows that queries made by users of the client applications remain fast. With the **help of the explain query**, identify slow queries, or queries that may become slow as the database grows and **add the appropriate indexing strategies** to ensure that the queries remain fast for users.

Part 4: Transactions and Concurrency [5%]

As Letterboxd grows, multiple users will simultaneously interact with the database, creating potential concurrency issues. Analyse the queries you implemented in Part 2 and identify which operations require transaction management and concurrency control to maintain data consistency. Choose and justify the isolation level used in each case with a comment.

Part 5: Security and Privacy [15%]

Keeping user data secure is of the utmost importance for Letterboxd. Implement the following security measures to help keep users data secure:

User Roles and Permissions

Create and configure the following database roles:

- **Data Analyst:** only read data from the database to find trends, and insights
- **Content Manager:** can modify film metadata, cast information, and handle content issues
- **Admin:** full access for a select few trusted Letterboxd senior engineers like yourself

Include the SQL commands to create these roles in your `schema.sql` file.

SQL Injection

SQL injection is one of the most common and dangerous web application vulnerabilities. Write **prepared statements** to protect queries that take user input against SQL injection attacks.

Data Deletion

In the EU, when a user deletes their account with a service, you are required by law to delete all personally identifiable data relating to that user. Write a query to delete a given user, and ensure that when the user is deleted, all of their data is also deleted from the database.

Part 6: Extra Mile [10%]

This is an opportunity for you to design and implement a feature that Letterboxd currently doesn't have but that you wish it did. This could be:

- Private reviews
- Discussion threads
- Advanced user statistics
- Advanced search and filtering options

Presentation and Documentation [15%]

You and your partner will give a demonstration in class on the week of 8th December covering your database design, executing the queries that you wrote, query optimisation techniques, transaction and concurrency considerations, and security and privacy best practices.

Both team members must partake in the demo, and the presentation must be split evenly.

Submission

You will be required to make an individual submission by 12th December that should include:

- `report.pdf`: a rigorous design document of ~1000 words. Write about design decisions made, optimisations, the evolution of your project as you worked on it, scope and limitations of your work, potential future work, and what you individually contributed to the project. Write for a technical audience who has already taken this course.
- `schema.sql`: a list of the create table, create index commands
- `queries.sql`: a list of all of the queries, stored procedures, triggers, and views
- `data.sql`: any data queries required to seed the database with sample data
- `erd.png/erd.pdf`: Your entity-relationship diagram. This can be drawn using a tool like draw.io, dbdiagram.io or even hand drawn if you'd prefer, but entities, relationships and their types must be clearly shown.