Coursework briefing

This document explains the arrangements for the coursework, a group project in which you will develop a web-based information system using the MovieLens dataset, with all data stored in a SQL database that you need to design. Through this coursework you will develop your knowledge of database technologies and your skills with database development methods and tools. The project runs through the term and will be self-managed with weekly supporting lab sessions.

The things you have to deliver are a group online presentation of a working system and an individual report. Your coursework mark for this module is the weighted combination of the individual and group presentation marks as described later (all group members receive the same mark for that element). The coursework is worth half of the marks available on this module (50%).

The coursework should be performed in groups of four and groups should be self-formed by the end of the first week using the wiki on Moodle. We will allocate people to groups who aren't in a group.

Design brief

You are asked to design and build a system to achieve the outline use cases for the scenario given in this design brief. How you do it and what you produce in detail is your choice. Essentially you get marks for what you build and the more sophisticated it is the more marks you can potentially get. Your work will achieve the threshold for a reasonable pass mark if you produce a system that implements each of the requirements in at least some form.

The system is for use by marketing professionals in the movie industry. It will make use of the MovieLens data which consists of information about movies, ratings and tags of movies provided by viewers, etc. The system is intended to enable marketing professionals to analyse how audiences have responded to films that have been released, and to help them understand the market for films that they are planning. It is intended to help them better understand the different kinds of viewers of movies and their varying preferences. The core functions that your system should provide are these:

Requirements/Use Cases to implement:

1. Visual browsing of the films dataset by title and date, and by genres and by ordered ratings. Manipulation of visual listing, e.g., to find the most popular action movie since 1995 etc.
2. Searching for information on a specific film in the dataset including its content, date, director, lead actors and rottentomatoes rating.
3. Analysis of viewers' reaction to a film in relation to their rating history (e.g., did people who tend to give low ratings give this film a low rating too, etc). Analysis of viewer ratings by ratings history and genres, e.g., do people who tend to rate romcom films highly also rate this film highly?
4. Tag data analysis. Can the film dataset be segmented by tag data. What correlations exist between the tag data, genres and rating data? Do individual viewers apply the same tags to different films and to different films in the same genre?
5. Predicting how a film will be rated after its release from the reactions of a small preview audience. Taking a sub-set of the viewers for a particular movie in the data set and treating them as though they were people at a preview, is it possible to predict the aggregate ratings of that film by all viewers in the dataset?
6. Analysing the personality traits of viewers who give a film a high or low rating. Using the personality/ratings dataset from GroupLens, examine whether there is a correlation between personality traits (e.g., extrovert, etc) and ratings for the 12 films in the Personality 2018 dataset. Using the genres data from the small dataset, is there a correlation between personality traits (extraversion, etc) and preferences for particular genres of film (horror, etc)? Note that to join with the genres data, the personality dataset and the small data set can be joined on filmid but not on userid.

Groups should make use of the small dataset http://files.grouplens.org/datasets/movielens/ml-latest-small.zip. For at least one of the use cases you will need to extend your database with additional movie details available from IMDB.

For Use case 6, there is a separate dataset of personality data made available by GroupLens. This dataset and the small dataset both refer to films using the same movie_id, which therefore allows the two datasets to be joined. Joining the datasets allows you to associate movie genre and tags with the ratings given in the personality data set. The users in the two data sets cannot be joined due to the hashing used on one. Also, note that the personality dataset resulted from a research project that used and produced data which are not needed for this coursework. Specifically, in the personality table, you will only need columns A to F; these give a score (from zero to 7) for each of the so-called 'big five' personality traits (openness, etc) for each user.

Groups should build their system so that it can run in a collection of Docker containers, using a container for each component, and the containers communicating via a private docker network. For example, one will be running the web server, another the database, another performing the data processing, etc. Using Docker will make it easier to develop on your local development machines, and to ensure that everyone in your group has the same versions of the software being used. You do not need to install the software you need directly onto your local machine, use containers instead.

See the Docker documentation here: https://docs.docker.com for detailed information about using Docker, or follow one of the courses on LinkedIn Learning (e.g., the Learning Docker or Docker Essential Training courses).

Visual Studio Code (VSC, https://code.visualstudio.com) is an excellent tool for developing web and database applications. In particular, investigate the Remote Containers extension, as this integrates VSC with Docker containers and makes developing with containers nice and straightforward. See https://docs.microsoft.com/en-us/learn/modules/use-docker-container-dev-env-vs-code/ for more information (this is on Microsoft Learn, you can login using your UCL account and get full access – this is strongly recommended as well). Alternatively you can use the IntelliJ IDEA IDE as this also supports container based development well.

It is up to you which programming languages, software packages, libraries, databases, etc. you make use of. Everything you need will be available as Open Source software, so there should not be any licencing issues, and there will almost certainly be a range of pre-defined official Docker images available at Docker Hub (https://hub.docker.com) for anything you use. For example, the official MySql image can be accessed at: https://hub.docker.com/_/mysql.

A core part of your system will be the database, which *must* be a SQL database, and should have an appropriate number of tables. Groups will need to create an entity relationship diagram (ERD) which they will translate into the schema for their relational database. The schema should be normalised.

You must design and implement the SQL database directly following the ERD, along with all the required SQL code to run queries on the database. You cannot use automated tools to create the database using a specification language, object-relational or similar mapping frameworks, or anything that generates SQL code automatically. You can write SQL scripts to create and configure your database (this is standard SQL code), and help manage the database.

NOTE: this is an important constraint – you *must* design and implement the database and SQL queries yourselves, and cannot use tools that generate the database and queries automatically. Your code must construct the SQL queries and run them, and extract the data from the result sets.

You have the choice of which other web, front-end and back-end technologies you use, but bear in mind that the primary goal is the design and implementation of a robust database so don't get side-tracked by trying to use the latest and greatest application frameworks!

You should not be using machine learning methods for this coursework, even though it might be possible and even quite interesting to do so. This coursework is focussing on the database design and data processing aspects rather than data analytics. You should therefore not be using Python ML libraries; you should be using SQL queries, extracting the data and writing the data processing code needed for the use case.

Your group should have a GitHub repository, used properly, with regular commits. It should be possible to build and deploy your application by pulling the latest version from the repository. The proper use of GitHub will be considered a core activity for the project work, if you need a refresher on using GitHub follow the tutorial on the GitHub Docs website (https://docs.github.com/en) or one of the courses on LinkedInLearning.

A well-designed application will have a gateway service connected to the internet, with all the other components and services running in a virtual private network (the Docker container network). The security of the application should be addressed and you should identify and implement strategies to keep it secure. For example, as you are using SQL the application should not be susceptible to SQL injection attacks.

Credit will be given if your application design can address scalability, so the application could potentially scale-up to thousands of simultaneous users. This might mean having multiple database servers, using caches, micro-services, and other strategies. Identify scaling strategies that minimise cost while still giving flexibility, and how to control costs. It is better to have an application that has clearly defined upper performance or scalability limits than one which simply consumes resources in an uncontrolled way.

Your application should have a reasonable web-based user interface but the focus of the work and our assessment of it is the back-end and the database design and implementation.

Project Deliverables

1. Group demonstration (75%).
   We will arrange and record a 15 minute online demonstration of each group's system in the final week of term. The demonstration should clearly show each working feature you have implemented from the list given earlier. Make sure that each feature is shown one-by-one in sequence, not all mixed up! And that each feature is introduced so it is clear what you are describing at any given time in the demo. You do not have to create a formal presentation with slides, but do aim for conciseness and clarity. No separate mark will be given for the demonstration. You will need to give us access to your GitHub repository.
2. Individual report (25%).
   Each group member should submit a report giving their own account of the design of the system and explanation of how the function of each use case is achieved, and an evaluation of the development strategy used (e.g., use of containers). The length of the main report content should be a maximum of 4 pages, plus any appendices as extra pages. This is individual work and should not be written in collaboration with the other group members.

The report should include a critique of how well the system achieves each use case, and give a summary of the work you personally did in each week of term towards the system design and build. The report may include in its appendices your ERD, tables, queries, and other relevant products of the project. You should also include the URL to your GitHub repository so that both the final version of the application and the history (commits) can be viewed. You can make the repository private provided that the markers are given access.

Marking

The allocation of marks will be according to the rubric table below. Marks will be returned using Moodle's marking rubric tool using this table format.

(Group) Demonstration of use case 1	7 marks
(Group) Demonstration of use case 2	8 marks
(Group) Demonstration of use case 3	12 marks
(Group) Demonstration of use case 4	15 marks
(Group) Demonstration of use case 5	15 marks
(Group) Demonstration of use case 6	18 marks
(Individual) a) Account of database design and SQL	6 marks
(Individual) b) Explanation and critique of use cases	7 marks
(Individual) c) Review of design effectiveness and systems operation	6 marks
(Individual) d) Record of personal contribution	6 marks

The marks for the group demonstration will be awarded equally to all members of a group participating in that demonstration. Marks will be awarded for the demonstration in relation to each use case and will take account of the extent and sophistication of what you use and create. The marks for each use case will consider the design of the overall application in terms of components, software choices, level of data processing and predictions achieved. We will consider the value to users of the system and its likely operational effectiveness, including features like the relative scalability (if attempted) and how security has been addressed.

The sections in the individual report should cover:

(a) The database design including a discussion of the state of normalisation of the database schema, and explanation of two of the most extensive SQL queries used. Your discussion should refer to the ERD created by your group collectively which you should include in the appendix of the report.

(b) Explaining the high-level system function for each use case and the approach taken, and your critique of how well the system achieves each use case.

(c) Your account of the effectiveness of the system design in terms of the likely value to users and of systems level operational features like scalability and security.

(d) An account of the work personally contributed by you each week towards the system design and build.

The content should be clear and concise, using appropriate terminology, making use of appropriate objective criteria (i.e., quantifiable) to describe and evaluate the work.