COMP0178: Database Fundamentals (PG – Term 1)

Coursework briefing

This document explains the arrangements for the coursework, a group project in which you will design and build an auction system. Through this coursework you will develop your knowledge of database technologies and your skills with database development methods.

This project runs through most of the term and counts for 50% of the marks available for the module. The deadline for submission will be listed on Moodle. Each week (until week 7) we will provide a lecture-format tutorial to explain the technologies and methods you will need to use in your projects. The project is self-directed in the sense that you will decide how you will achieve the requirements that we set for you in the design brief below. You will likely need to investigate additional techniques needed to implement your systems.

The coursework must be performed in groups of four and groups must be formed by the end of the second week. By the end of the second week, your group needs to have met and discussed the peer assessment scheme you will use. A list of students taking this module is posted on the Moodle page to help you form groups, and an example peer assessment scheme you are welcome to use unaltered is also uploaded. Please submit a document containing a list of your group members and your peer assessment scheme on Moodle. Students who haven't joined a group by the end of the second week of term will be allocated to groups by the tutors.

Each group will build a functioning database system that achieves the capability requirements set out below in the Design Brief. You should build your project using XAMPP (https://www.apachefriends.org/) or a similar tech stack suitable for your OS (MAMP for OSX or Windows; WAMPServer for Windows; other options available). XAMPP is a multiplatform software that provides web server services on localhost using Apache, MySQL (MariadDB), and PHP, and is installable on UCL lab machines. However, we recommend that you install XAMPP (or similar) on your own machine. You may not use any proprietary development frameworks. You will need to populate your database with illustrative data.

You *may* host your project on Microsoft Azure to gain experience of hosting a real service and managing a cloud-based MySQL instance. Azure supports PHP with MySQL or MariaDB integrated with version control services such as Github. This allows groups to develop their systems on their local or lab machines with their *AMP stack and then push their code into the cloud to run on Azure. However, no support will be given for cloud / Azure tools by the tutors.

Your design work should be focused on the database design and follow the structured methods taught in class. You should translate the requirements as you interpret them into what data should be stored, the transactions those data support, how the data will be modified, and who will have access to it. You must create an entity relationship diagram to fully represent the data of interest and their relationships and attributes. You should translate that diagram systematically into a database schema defining the tables and relationships for your database. You should show that the database is in third normal form.

To start you off, the code for a partly functional website has been provided. The main things missing are a database and PHP functions for interacting with that database. The starter code uses the free and open source frameworks Bootstrap (CSS-based) and jQuery (JavaScript-based).

Submission requirements and assessment

This project has two parts: a group submission (submitted once per group) and an individual peer assessment, which each member of the group is responsible for submitting.

Group submission

The deliverables for the group submission are (1) a video of a demonstration of your group's working system, (2) a design report, and (3) your group's source code.

The video demonstration should be about 5 to 8 minutes long with voice-over narration. The video must show explicitly how each capability listed in the design brief below is achieved in your system. It must go through each capability in the order in which it is listed in the Design Brief (extra functionality for core features can be covered when going over those features or separated into its own section). The narration must refer to the capability and preferably the video should flash back to the visual list. With a spoken narration the video must show each capability you have achieved either through interaction with the user interface of your system or through execution of the relevant queries on the database through phpmyadmin. Examples of videos from previous years are on the Moodle page. Videos should be uploaded to Youtube and made unlisted.

You need to submit your design report (a single pdf file only) which should include (among other things) the link to your video. The submission link is on the Moodle page. The design report should contain (in this order):

1. URL for your Youtube video.
2. Your entity relationship diagram, giving any assumptions that it makes about the processes that use the data.
3. A listing of your database schema (list of table names and attributes) with an explanation of how it translates the ER diagram.
4. An analysis showing that the database schema is in third normal form.
5. A listing of your database queries accompanied by very brief/high-level descriptions of what each does/is for (preferably grouped by the file they appear in or the system functionality they support).

Your report does not need an introduction or conclusion, just the elements listed above.

Additionally, you should submit an archive containing your source code (the PHP, SQL, HTML, CSS, etc. files needed by your site).

- Please only submit your source code -- no external/prewritten libraries or dependencies and no large datasets (such as large folders of images).
- Please put your group number in the zip file name.

You should submit a draft entity relationship diagram as a separate submission at the end of the third week of term. The diagram will be the best view you have at that point even if it changes later. The diagram won't be assessed but we will give you formative feedback on it.

Marks for your project work will be awarded (i) for the capabilities (i.e. functional requirements) your system achieves, and (ii) for evidence of the database design process you followed. Total marks for these two components will be allocated on an 80%/20% basis.

Individual submission

Each student must prepare a PDF containing (1) a self-assessment, and (2) peer assessments of each of the other members of their group. This document should be uploaded to the coursework submission box on Moodle. One member will therefore have three submissions (two group submissions (report containing video URL; codebase) and one self/peer assessment submission) while all other members will have one submission (self/peer assessment only). Your mark for the group submission will be modulated by the marks given by your groupmates to yield your final coursework mark.

Design brief

Your online auction systems can have as many of the capabilities listed below as you are able to develop. Functionality is separated into core functionality needed to operate the auction site and extras that can be implemented on top of it. Marks will be awarded for each capability achieved and will take into account the quality and completeness of the design and implementation. You should design a database schema to store the data needed to achieve these capabilities. You should build the database and populate it with illustrative data. You should develop the queries and updates needed to achieve each capability. The presentation/design of your user interface and its usability is NOT being assessed and you should avoid spending time on these; our concern is with the design of the database, the queries on the database to achieve the capabilities and the application function.

Requirements list:

Component	**Description**	**Mark %**
Design	Evidence of a sound database design process • ER diagram • Table design (schema) and normalization o 0 pts: Absent. o 5 pts: Basic response with flaws/misunderstandings. o 8 pts: Mostly correct with minor flaws. o 10 pts: Strong answer.	**20** (10) (10)
Core functionality:		
1	Users can register with the system and create accounts. Users have roles of seller or buyer with different privileges.	**10**
2	Sellers can create auctions for particular items, setting suitable conditions and features of the items including the item description, categorisation, starting price, reserve price and end date. o 5 pts: Basic functionality partially implemented. o 7-8 pts: Basic functionality implemented with minor issues. o 10 pts: Basic functionality with strong implementation.	**10**

3	Buyers can search the system for particular kinds of item being auctioned and can browse and visually re-arrange listings of items within categories.	**15**
4	Buyers can bid for items and see the bids other users make as they are received. The system will manage the auction until the set end time and award the item to the highest bidder. The system should confirm to both the winner and seller of an auction its outcome. o 8 pts: Basic functionality partially implemented. o 10-12 pts: Basic functionality implemented with minor issues. o 15 pts: Basic functionality with strong implementation.	**15**
Extra functionality:		
E1-4	Extra functionality related to core features requiring usage of a database. o 0-20 points depending on what has been implemented.	**20**
E5	Buyers can watch auctions on items and receive emailed updates on bids on those items including notifications when they are outbid.	**5**
E6	Buyers can receive recommendations for items to bid on based on collaborative filtering (i.e., 'you might want to bid on the sorts of things other people, who have also bid on the sorts of things you have previously bid on, are currently bidding on). o 0-5 points depending on what has been implemented.	**5**