The goal of the project is to develop a functional database-backed application from design through implementation. The project must include the following.

- **A front-end graphical user interface.** The interface can be implemented as a web application, a mobile application, or a local application.

- **Transaction processing.** Through user interactions, your application must require basic transaction support that includes inserting new rows, updating existing rows, and removing rows. Your application must also require the use of basic queries against a database, e.g., as part of loading page contents, retrieving information about items, and so on. Note that no SQL statements or output (that isn’t formatted) should be a part of the user interface.

- **Faceted search.** Your application must allow users to search for items through the interface and the results of such searches must be displayed. The searches must allow users to specify various types of information and/or ranges (similar to a “search filter” found on many web pages and mobile applications). Search results must be formatted (as opposed to just displaying the returned table) and interactive (i.e., users can select a result for further information).

- **Analytics.** The application must include analytical views of the data (in some fashion) involving the ranking of items (e.g., most popular to least, most purchased to least, etc.), summary statistics using grouping (e.g., average cost by type of item), and so on. This will likely be the most difficult part to “naturally” incorporate into your application. One approach would be to include some form of “administrative” dashboard (page).

Additional requirements are provided below. The project will be broken into a series of steps (deliverables) to be done over the remainder of the semester. Each step is described further below. Note that this is an individual project and each student project must be different.

**Step 1: Proposal (due Tues, 11/1).** You must provide a description of the application you plan to develop as well as the technology you plan to use (e.g., PHP, NodeJS, Java w/ Swing, Python w/ tkinter, etc.). Your proposal can be at most 1-page in length. Your description should describe the general theme of your application along with the main user functionality it will support. Note that you can develop an original application or develop a “clone” of an existing application. Your application must be large enough in scope to require at least seven tables, three of which must have foreign keys (i.e., “relationship” tables). At this point, it is better to be ambitious in scope and then reduce it later as needed. For this step turn in your proposal in class on the due date.

**Step 2: Page Design and Layout (due Tues, 11/8).** You must provide an initial design of the user interface. The document must include: (a) a brief summary of the application, (b) sketches of how each page will be laid out; and (c) a description of the interaction among pages (how the pages can be navigated). If you are planning on cloning an existing site, you should include a description of how your design will differ. For this step turn in your design and layout in class on the due date.
Step 3: ER Diagram (due Tues, 11/15). Based on your user interface design, you must create an ER diagram representing the information you will be storing in your database. Your ER diagram must contain entities, keys, attributes, relationships, cardinality constraints, and weak entities as appropriate. For this step turn in (a) a brief description of the project, (b) any changes/updates you’ve made, and (c) your ER diagram in class on the due date.

Step 4: Table Designs and Data Population (due Tues, 11/22). Create a revised ER diagram as needed and then use it to create your initial database tables. Your tables must be implemented using CREATE TABLE statements in a separate script file. In addition, you must populate your tables with enough data to help you with testing (e.g., 30 or more rows per table). Your INSERT statements must also be stored, separately, in a script file. Note that you might consider creating a basic program for automatically generating data. There are also some tools online that can be used to generate “fake” data. Turn in a print out of your revised ER diagram and the two scripts in class on the due date.

Step 5: Status Update (due Thurs, 12/1). Write up a brief status update of your project implementation. Include screen shots of what you have working. Turn in your update in class on the due date.

Step 6: Project Implementation and Writeup (due Wed, 12/14). Your project is due on or before the day of the final. You must submit all of your (well commented) code and final ER diagram to GitHub (instructions will be provided separately in Piazza). In addition, you must create a screen recording presentation that must include in the following order: (1) a brief description/overview of the purpose of the application; (2) a demo of the application itself including all features implemented; (3) a walkthrough of your professionally drawn ER diagram (no hand drawings, must look professionally created); (4) a walkthrough of your tables; (5) a walkthrough of the more difficult queries used in your application; (6) how you tested your application; (7) a list of items that aren’t working and/or challenges you had; and (8) next steps if you were to continue developing the application. You must include either the video or a link to the video in your GitHub repository. Your recording should be around 8-12 minutes in length (but shorter is fine as long as you cover all the points above). If you use slides as part of your presentation (recommended), you must include these in your GitHub repository as well. Finally, you must include instructions on how to install and run your application.

Grading. The project is worth a total of 60 points. Steps 1-5 will be worth 3 points each. The remaining points will be based on your final implementation and demo video described in step 6. Note that your project will be graded on the following criteria.

- **Quality.** The overall quality of the code, queries, and database design. Whether your application contains obvious bugs or implementation issues.

- **Completeness.** Whether the application is complete, with each screen working, screens finished, and the application thoroughly tested and debugged.
- **Difficulty.** The overall scope of the project, the amount of effort required to develop the user interface, and the complexity of the queries and functions tackled.

- **Documentation.** Updated ER diagrams, clear and easy-to-follow instructions for running the application, and code is commented and well formatted.

- **Presentation.** Covers each aspect asked for, is clear and professional, and is well organized.