The goal of the project is to develop a functional database-backed application from design through implementation. Your application can be original (a product idea you come up with) or can be a “clone” of an existing application. The application must include the following.

- **A user interface.** The interface can be implemented as a web application, mobile application, or as desktop (stand-alone) application. As a desktop application, you can implement a graphical user interface or a text-based user interface.

- **Data persistence.** Your application must provide data persistence using a backend database. Users (via the user interface) must be able to add, modify, and remove underlying data via your application’s user interface. Users must also be able to explore and search for underlying data stored in the database. Finally, your application should have a significant amount of data “pre-populated” in the application (which can be “fake” data).

- **Transaction processing.** Your application must support transaction processing operations (triggered by interactions with the user interface) that result in inserting, updating, and removing underlying data rows. Your application must also perform basic queries against underlying data, e.g., as part of loading page/screen 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 user 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 be interactive (i.e., users can select a result to obtain further information).

- **Analytics.** Your application must include different analytical views of relevant underlying data. The analytical views must include different “conceptual” rankings of items (e.g., most popular to least, most purchased to least, etc.) as well as multiple relevant summary statistics using grouping (e.g., average cost by type of item). Analytics support will likely be the most difficult part to “naturally” incorporate into your application. One approach would be to include some form of “administrative dashboard”.

Additional requirements are provided below. The project will be broken into a series of steps 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 Thurs, 10/5). 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 does not need to be more than 1-page in length. Your description should include the general theme of your application along with the main user functionality it will support (i.e., what can users accomplish using your application). Your application must be large enough in scope to require at least seven tables, at least three of which must have foreign keys (i.e., “relationship” tables). Your application must also have at least two many-to-many relationships (where items can be related to many other items, and vice versa). At this point, it is better to be ambitious in scope and then reduce it later as needed. For this step turn in your typed proposal in class on the due date.

Step 2: Interface Design and Layout (due Thurs, 10/19). You must provide an initial design of your application’s 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/screens (how the pages/screens 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 Thurs, 11/9). 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: Status Update (due Thurs, 11/30). After Step 3, you should begin implementing your table designs, populating them, and developing your application’s user interface. For this step, you must 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 5: Project Implementation and Writeup (due by Fri, 12/15). Your project is due on or before the last day of finals week. 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 video presentation of your work. The video must include: (1) a brief overview of the application you developed; (2) a demo of the application itself including all features implemented; (3) a walkthrough of your ER diagram (no hand drawings, must look professionally created); (4) a walkthrough of the more complex queries used in your application; (5) a description of the approach you used to populate your application with data; (6) an overview of 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 upload your video to YouTube and include a link to the video in
your GitHub repository. *Your recording should be no more than 10 minutes in length (but
shorter is fine as long as you cover all the points above).* You must develop slides as part
of your presentation, which you must also include in your GitHub repository. Finally, you
must include instructions on how to install and run your application.

**Grading.** The project is worth a total of 70 points. Steps 1–4 will be worth 4 points each.
The remaining points will be based on your final implementation and video presentation
described in step 5. 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.