# Overview

In assignment 3, you built a static website using publicly available data sources from data.gov. In this assignment, you are asked to build out the concept as a live data-driven website, working with live data, using a database to store data and using full-stack application development technologies to create an interactive, data-driven website. The site is to be deployed to the cloud. The site should implement all CRUD operations and show at least one database-driven master-detail (1-many) relationship. This means that there should be at least one set of items on the site such that clicking the item takes you to a page with more data related to that item. For example, if you have data on natural hazards, perhaps the data includes the city or state affected by the event. If so, the city or state can be a hyperlink and clicking on the city or state could take you to a page showing all events in the database that affected the city or state.

The assignment will give you the opportunity to develop familiarity with MVC applications, database connectivity, cloud deployment, charting, and in general gain confidence in your developer capabilities.

# Activity

Implement your static application from assignment 3 as a data-driven web application that performs something useful for users interested in the domain you have picked, and deploy it to the cloud. The application should show at least one master-detail relationship and should have between 2-3 pages. There are no specifications on the number of database tables, but the recommendation is to keep things as simple as possible and use perhaps 3-5 related tables in the database. Deploy the web application to a cloud service. Azure should be the easiest to use, given our use of C# in this class.

Try to make the assignment as detailed and professional as you can make it. Assignments 3 and 4 are attractive projects for you to demonstrate your skills to prospective employers.

To complete this assignment, you are required to complete the following tasks:

1. Deploy a web application that implements your business service over the web. Deploy it to a cloud platform, so it can be accessed worldwide
2. To develop comfort with working with APIs and Databases, your application should consume at least 1 API end point.
3. Add an about us page to the application, as specified in assignment 3. You may however, want to supplement the page with a section titled say, “technical details” and quickly summarize the things you would like to “show off” to potential interviewers.
4. All grading should be possible from a review of the application behavior. Students should not expect a code review on GitHub to confirm functionality.

The assignment will be graded as follows:

Database design and use: The database should have at least 3 related tables. Data in at least one of the tables should be populated from an API. There should be at least one use for each CRUD operation (Create, Read, Update, and Delete). These CRUD operations should be relevant to the application, i.e. the user should be able to add, read, update and delete data that is relevant to the application.

API use: The application should pull and parse data from at least one API as specified in the project. This data should be used to populate at least one table in the database.

Master-detail relationship: A master-detail relationship is essentially a one-many relationship and your application should have at least one master-detail navigation. This means that your application should have a page that lists items in the master table (the table on the one side of the relationship). The list should include hyperlinks or buttons that take the user to the details page (many side of the relationship) related to the item. For example, if your application has a one-many relationship between university departments and courses offered by each department, you could have a master page that lists departments and their properties (e.g. contact information etc). Clicking on the department’s name would take you to a page that lists all the courses offered by the department.

Look and feel: The application should have a consistent and professionally reasonable look and feel, that is appropriate for a professional demo to an employer. The project is about functionality and not about look and feel, however, projects that do not look presentable will hurt students if demonstrated to an audience. Therefore, develop a simple and presentable layout for your site and use the layout throughout the site. Keep all look and feel consistent and presentable.

Chart: The application should include at least one chart that is driven by the data pulled from the API. Typically this chart will use a standard charting framework such as chart.js

About us: This page should follow the specifications of Assignment 3. The ERD diagram and API endpoints should be updated if necessary to reflect the database design actually implemented in the project. The git log output should be updated. If you feel good about your project, it may also be a good idea to add a brief paragraph with talking points about the technical features of the project. This can be a helpful reminder if you demonstrate the project to friends, recruiters etc.

Self-reflection: Each team member should include a self-reflection as a comment to the assignment on Canvas. This information is very important for us and is the primary inputs we use to continuously improve the assignment for future semesters. This information also helps us identify any thing that needs to be discussed further in class. The submission section below specifies more details about the self-reflection.

# Submission

Submit the URL to your application on the cloud.

Also, each member in the team should write a self-reflection about the project as a comment to the assignment. The self-reflection should include an estimate of the total time the student spent on the project, what you learnt from doing the project, how you managed your time and team coordination while working on the project and anything the instructor can do to improve the project experience for students in future semesters.

# Recommendation

More adventurous students can try to implement a Netflix-queue type capability where users can number items in the queue and on form submission, the items are displayed in the selected sequence.

Once the assignment is graded, we would highly recommend that you stop all services associated with the project on the cloud service. This will conserve your free credits on the cloud service so you can use them when you turn the project back on while applying for internships and jobs.

We also strongly recommend using layouts to obtain a consistent look and feel for the application.