# CSC372: Web Development

## Individual Assignment 6: Jokebook

**Due: April 18th, 2025**

For this assignment, you will build a full-stack application with a Node/Express backend.

Given the following JavaScript objects, use SQLite to design an appropriate relational database to store the object data.

```
        let categories = ['funnyJoke', 'lameJoke'];
        let funnyJokeList = [
          {
               'setup': 'Why did the student eat his homework?',
               'delivery': 'Because the teacher told him it was a piece of cake!'
          },
          {
               'setup': 'What kind of tree fits in your hand?',
               'delivery': 'A palm tree'
          },
          {
               'setup': 'What is worse than raining cats and dogs?',
               'delivery': 'Hailing taxis'
          }
        ];
        let lameJokeList = [
          {
               'setup': 'Which bear is the most condescending?',
               'delivery': 'Pan-DUH'
          },
          {
               'setup': 'What would the Terminator be called in his retirement?',
               'delivery': 'The Exterminator'
          }
        ];
```

1. Firstly, implement a web service (CRUD API) using Node/Express.

Follow the MVC architecture pattern with a model, a router, and a controller to provide the following API endpoints:

i. Jokebook Categories: GET endpoint at /jokebook/categories that:

- responds with a list of possible categories in the jokebook database.

ii. Jokes in a category: GET endpoint at /jokebook/joke/:category that:

- responds with the list of jokes from the specified /:category
- add an optional query parameter 'limit' that limits the number of jokes returned.
- If the category is not valid, respond with an appropriate error message.

iii. Random Joke: GET endpoint at /jokebook/random that:

- responds with one joke from the database chosen at random.

iv. Add a new joke: POST endpoint at /jokebook/joke/add that:

- Requires three parameters: category, setup, and delivery, sent within the body of the request.
- Should add the joke to the relational database.
- Responds with the updated jokebook for that category.
- If missing one of the required parameters, respond with an appropriate error message.

Use ThunderClient to test all the endpoints.

2. Finally, implement a view that displays the jokes on a webpage. Style and appearance are up to you. You may choose one of the following options for your views:

1. Client-Side rendering with JavaScript and fetch().
    a. Please do not mix JavaScript and HTML: “onClick” or “innerHTML” are forbidden.
2. Server-Side rendering (change data return types from JSON to view names)
    a. with Pug.
    b. with EJS.
3. Client-Side Framework: React

The landing page should display a random joke from the database.

Provide functionality for the user to view a list of all the joke categories by calling the first endpoint. Selecting one of the returned categories should then call the second endpoint that gets the jokes of a category. This must also be possible by directly inputting a category name on a search field. Finally, add a form that lets a user create a new joke by calling the last endpoint. After this succeeds, display the jokes in that category to show it has been added.

EXTRA CREDIT: *If a user searches for a category that is not any of the above, add functionality to search for the category from the https://sv443.net/jokeapi/v2/ API. Search for a maximum of 3 two-part jokes for such a category. No inappropriate jokes at any point please. You will then add the new category and the list of 3 jokes to your local jokebook database.*

**To submit your work, please use a new public GitHub repository, which you can name “Jokebook”. Submit a link to the new repo on Canvas.**

> ***Do not use your github.io repository, since GitHub Pages has no Node runtime environment. Additionally, your express server file must be at the root folder of the project for npm scripts to be executed, so a subfolder would not work well.***

**Please do not upload your node modules folder to source code management. You may add a .gitignore file at the root directory that will ignore the whole node_modules folder. Make sure your package.json lists all the dependencies for the needed modules.**