# Final Project

Start Assignment

- Due May 13 by 11:59pm
- Points 100
- Submitting a file upload
- File Types zip, mp4, and txt
- Available Apr 28 at 11:59am - May 13 at 11:59pm

## Final Project

---

### DUE: Monday 05/13/24 - 11:59PM
### NO Extensions. NO Late Submissions.

---

Working with your Final team - create an Express.js Server Application to demonstrate your understanding of Node.js Servers, Express Router, and Database Integration. The Server Application must interact with your selected API from the Midterm and store data on a Mongo Atlas Cloud Database.

---

### Express.js Server Application Requirements

- Create a Express.js Server Application similar to the Deck of Cards Server Application
- Utilize Express.js Router and integrate with MongoDB
- The FInal Project should have the following structure:
  - routes folder:
    - `search.js (see details below)`
    - `history.js`
  - services folder:
    - `api.js`
      - Service for interacting with the selected API
    - `db.js`
      - Service for connecting to and handling CRUD operations with MongoDB (use HW3)
  - `package.json`
    - Properly filled out with project details

- `server.js`
  - Contains server logic such as routing, database connection and server start
- `.env`
  - File for storing environment variables

---

## search.js

### 1. Endpoint `GET /search`

- This endpoint **accepts a query parameter:** `searchTerm`
  - This endpoint should searches the selected API by the provided search term
  - The endpoint should compose a minimal JSON response results array
  - The results array should only contains two keys:
    - one key should represent the unique identifier
    - one key should represent display text related to the search result

- This endpoint should also create an entry in the `search_history` collection in MongoDB
  - If this search term does not exist in Mongo DB then create a new document to be saved into your Mongo DB `search_history` Collection
  - If this search term does exist in Mongo DB then update the document with the a new date for `lastSearched`

```
EXAMPLE DOCUMENT
{
   "searchTerm"     // (String) the search term the user entered
   "searchCount":   // (Int) the matching result count from the API
   "lastSearched":  // (Date) the date/time the last search was performed for the given
keyword
}
```

### 2. Endpoint `GET /search/:id/details`

- This endpoint **accepts a path parameter:** `id`
  - This path parameter represents the user's selection and some **unique dynamic identifier** that is related to your API
- This endpoint **accepts an optional query parameter:** `cache`
  - This query parameter represents whether the user wants to use the cache or not

- This endpoint retrieves detailed data for the selected item based on the cache option
  - If the cache option is set to false (default):
    - Retrieves the selected item by identifier from your API
    - Saves an entry in the `search_cache` collection in MongoDB
  - If the cache option is set to true:
    - Attempts to find the selected item in the `search_cache` collection in MongoDB and returns it if found.
    - If the item is not found in the `search_cache` collection, retrieves the selected item by unique identifier from the API.
    - Saves an entry in the `search_cache` collection in MongoDB

---

## history.js

### 1. Endpoint `GET /history`

- This endpoint retrieves data from the `search_history` collection in MongoDB
- This endpoint accepts and an **optional query parameter:** `searchTerm`
  - If no query parameter is provided, returns a JSON response of all search history saved in the Atlas Cloud MongoDB.
  - If a query parameter is provided, returns a JSON response of the search history associated with the search term in the Atlas Cloud MongoDB.

---

### What to Upload to Canvas

- ZIP File with:
  - server.js
  - package.json
  - .env
  - routes
    - search.js
    - history.js
  - services

        - api.js
        - db.js

- Video Presentation (max time 25 minutes) using Screen Sharing
    - Upload a either the mp4 video or a ,txt file with a link.
        - The .txt file should only contain Zoom Cloud Recordings, Team Meetings or Canvas Videos.
            - NO third party URLs will be accepted
    - Video Presentations can be easily created by Using Zoom and recording a meeting to your local machine or to the cloud.

Each member of the team **MUST** take the Team Evaluation Survey

- This quiz is 10% of the Final Project Grade.
- Taking the quiz is mandatory, and failing to do so will result in a 0 for this portion. However, simply completing the quiz will automatically award the full 10% of the grade.
- This quiz is a feedback on your team members performance and contribution.

---

**Final Project Presentation:**

- As a team, create a video presentation to present your application in 25 minutes or less.
- Collaborate to create a coordinated presentation that is organized and focuses on the Final Project code (slides not required).
- Run a demo of your Final Project using a REST CLIENT
- Each team member introduces themselves and presents on a specific portion of the code written, covering:
    - `server.js`, `search.js`, `history.js and db.js`
    - not api.js as this was presented in the Midterm
- Each team member must discuss and explain code from one of these files:
    - Explain the code flow, how the code works, and how it relates to achieving the application's functionality.
    - Focus on presenting code aspects not extensively covered in class lectures or materials.
    - Ensure your code explanation is clear and concise.

**Final Project Code:**

- Code executes without errors during grading.
- Code includes all functionalities detailed in the provided requirements.
- Code follows proper ES6 syntax, utilizing let and const appropriately.
- Code is clean and well-formatted, with logical variable names and consistency in asynchronous handling.
- Avoids hardcoding wherever possible, except when necessary or mandated by the API.