

JULY 2025

# **Full stack development**

Gateshead College

Amarjit

# Assignments

**Assignment 01 - Create a Record format in JavaScript.**

**Assignment 02 - Creating a working Database/Record System.**

**Github repo**

<https://github.com/hey-am0/SkillsBootcamp-GC/>

# Assignment 01 - Record format in JavaScript

Using a web-based technology: Express/Node/EJS create information relating to your favourite tv/movie/radio shows (you choose) or your own topic within a record format.

You will need to add some additional information like: Title, seasons, episodes, year, cast, type, rating, poster, and directors etc..

Display the data/record on web browser by using either node, express or EJS, remember you need to show an images & code listing to show it is working.

# Assignment 01 - Record

## Batman Begins (2005)



- Type: Movie
- Director: Christopher Nolan
- IMDb Rating: 8.2
- Cast:
  - Christian Bale
  - Michael Caine
  - Liam Neeson
  - Katie Holmes
  - Gary Oldman

```
// Assignment-01
// Note: Requires Node/Express, EJS
//
const express = require("express");
const app = express();
const port = 3000;

app.set('view engine', 'ejs');
app.use(express.json());
app.use(express.static('public'));

// Parse JSON requests
app.use(express.json());

// Movie Record
const movie = {
  title: "Batman Begins",
  year: 2005,
  type: "Movie",
  seasons: null,
  episodes: null,
  directors: "Christopher Nolan",
  imdbRating: "8.2",
  poster: "https://m.media-amazon.com/images/...",
  cast: [
    "Christian Bale",
    "Michael Caine",
    "Liam Neeson",
    "Katie Holmes",
    "Gary Oldman"
  ]
};

// Routes
app.get('/', (req, res) => {
  res.render('index', { movie });
});

app.use((req, res) => {
  res.status(404).send(
    '<h1>404 Not Found</h1>' +
    '<p>The requested page <strong>4</strong> not found.'
  );
});

app.listen(port, () => {
```

## Sample code

- Record in JS
- GET Routes in Express
- Renders HTML via a ejs template

## Github link:

<https://github.com/hey-amol/SkillsBootcamp-GC/tree/main/assignment/assignment01>

# Assignment 01 - Route & View

```
app.get('/', (req, res) => {  
  res.render('index', { movie });  
});
```

Above is 1 example of a GET request in Express

On the right is the EJS template used to output HTML

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1">  
  <title><%= movie.title %> - Info</title>  
  <link rel="stylesheet" href="/style.css">  
</head>  
<body>  
  <h1><%= movie.title %> (<%= movie.year %>)</h1>  
  ">  
  <ul>  
    <li><strong>Type:</strong> <%= movie.type %></li>  
    <li><strong>Director:</strong> <%= movie.director %></li>  
    <li><strong>IMDb Ratings:</strong> <%= movie.imdbRating %></li>  
    <% if (movie.seasons) { %><li><strong>Seasons:</strong> <%=  
movie.seasons %></li><% } %>  
    <% if (movie.episodes) { %><li><strong>Episodes:</strong> <%=  
movie.episodes %></li><% } %>  
    <li><strong>Cast:</strong>  
      <ul>  
        <% movie.cast.forEach(person => { %>  
          <li><%= person %></li>  
        <% } %>  
      </ul>  
    </li>  
  </ul>  
</body>  
</html>
```

# Assignment 02 - Database/Record System

On this Task, you must create a working database record system, using any web-technology you feel comfortable with.

Main features, will include *adding*, *removing*, *deleting*, and *searching* (CRUD). The main point of this task is to demonstrate your capabilities of being a Full Stack Developer

## Assignment 02 - My notes/goals

To create a simple CRUD app that adds, edits, deletes, lists **movie** records

To install mongodb locally via homebrew

Used mongosh mongodb://localhost:27017 to run mongo locally

To use EJS for templating HTML

Try to use a simple callbacks/promise to simplify syntax

Use a sample record structure for quick insertion testing

**Github link:** <https://github.com/hey-amo/SkillsBootcamp-GC/tree/main/assignment/assignment02>

# Assignment 02 - Sample record structure

```
// Movie Record Structure (example)
const sampleMovie = {
  title: "Batman Begins",
  year: 2005,
  type: "Movie",
  director: "Christopher Nolan",
  imdbRating: "8.2",
  poster: "https://m.media-amazon
  cast: [
    "Christian Bale",
    "Michael Caine",
    "Liam Neeson",
    "Katie Holmes",
    "Gary Oldman"
  ]
};
```

This is a JSON object which holds a sample movie.

The app will allow the user to add, edit, delete, list movies using this structure



# Assignment 02 - Screenshots /1

## Node running

```
Node.js v24.2.0
@AD assignment02 %
@AD assignment02 % node app.js
(node:12900) [MONGODB DRIVER] Warning: use
has no effect since Node.js Driver version
[Use 'node --trace-warnings ...' to show w
App running on http://localhost:3000
```

## Mongodb running

```
2025-04-24T10:00:00.000Z mongod[77512]:
Current MongoDB Log ID: 486a738854656461c7a368d
Connecting to:
  mongod://localhost:27021/?directConnection=true&serverSelectionTimeoutMS=100
  &socketTimeoutMS=30000
Using MongoDB:
  8.0.10
Using Mongoose:
  8.6.3

For mongod info see https://www.mongodb.com/docs/mongodb-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically at
https://www.mongodb.com/docs/mongodb-shell/.
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2025-04-24T10:00:00.000Z: Access control is not enabled for the database. Read and write access
to data and configuration is unrestricted
2025-04-24T10:00:00.000Z: This server is bound to localhost. Remote systems will be unable to
connect to this server. Start the server with --bind_ip address to specify which IP addresses it
should accept connections from. To bind to all interfaces, use --bind_ip_all. If this message is
seen, start the server with --bind_ip 127.0.0.1 to disable this warning
2025-04-24T10:00:00.000Z: MFT TSM018: The open file descriptors too low

-----

test> use moviesdb
switched to db moviesdb
moviesdb> db.movies.insertOne({ title: 'test movie', year: 2023, director: 'test director' })
{
  acknowledged: true,
  insertedId: ObjectId('68b737384656461c7a368e')
}
moviesdb> db.movies.find()
{
  _id: ObjectId('68b737384656461c7a368e'),
  title: 'test movie',
  year: 2023,
  director: 'test director'
}
moviesdb> █
```

# Assignment 02 - Screenshots /2

## CURL - List all

```
root@AD ~ # curl http://localhost:3000/movies
{"count":2,"movies":[{"_id":"685a737354656d461c7a760a","title":"test movie","year":2023,"director":"test director"},{"_id":"685a875d492ae89f596fb733","title":"Batman Begins","year":2005,"type":"Movie","director":"Christopher Nolan","imdbRating":"8.2","poster":"https://m.media-amazon.com/images/N/MV5BODIyMDdhNTgtNDIwOCUzMiUwLWE2NDItODQ5MTdkNzY3ZTdhXkEyXkFqcGc@._V1_SX300.jpg","cast":["Christian Bale","Michael Caine","Liam Neeson","Katie Holmes","Gary Oldman"]}]}
root@AD ~ #
```

## CURL - Insert

```
root@AD ~ # curl -X POST http://localhost:3000/movies \
-H "Content-Type: application/json" \
-d '{"title":"The Matrix","year":1999,"director":"The Wachowskis"}'
{"message":"Movie created successfully","id":"685a87af492ae89f596fb734","movie":{"title":"The Matrix","year":1999,"director":"The Wachowskis","_id":"685a87af492ae89f596fb734"}}
root@AD ~ #
```

# Assignment 02 - Screenshots /3

## CURL - Insert sample movie

```
@AD ~ % curl -X POST http://localhost:3000/sample-movie
{"message":"Sample movie inserted successfully","id":"685a875d492ae89f596fb733",
"movie":{"title":"Batman Begins","year":2005,"type":"Movie","director":"Christopher Nolan","imdbRating":"8.2","poster":"https://m.media-amazon.com/images/M/MV5BODIyMDdhNTg1NDIhOC02MjUxLWE2NDI0ODA5MTdkNzY3ZTdhXkEyXkFqcGc@._V1_SX300.jpg","cast":["Christian Bale","Michael Caine","Liam Neeson","Katie Holmes","Gary Oldman"],"id":"685a875d492ae89f596fb733"}}
@AD ~ %
```

## CURL - Update

```
@AD ~ % curl -X PUT http://localhost:3000/movies/685a875d492ae89f596fb733 \
-H "Content-Type: application/json" \
-d '{"imdbRating":"9.0"}'
{"message":"Movie updated successfully","modifiedCount":0}
@AD ~ %
```

# Assignment 02 - Screenshots /4

CURL - Insert sample movie

```
qad ~ % curl -X DELETE http://localhost:3000/movies/685a875d492ac89f596fb733  
{"message": "Movie deleted successfully", "deletedCount": 1}  
qad ~ %
```

# Assignment 02 - Screenshots /5

## Listings page

### Movies

Total Movies: 3

Batman Begins (2005)

Director: Christopher Nolan

RDD Rating: 4.2

Type: Movie

Cast: Christian Bale, Michael Gough, Liam Neeson, Keri Holmes, Gary Oldman



Movie ID: 68f4a5d2c68f81f480c05a5b2

[View Details](#) | [Edit](#) | [Delete](#)

Indiana Jones and the Temple of Doom (1984)

Director: Steven Spielberg

RDD Rating: 4.5

Type: Movie

## Movie details page

[Home](#) > Movie Details

### Movie Details

Success: Movie updated successfully

#### test movie1

Title: test movie1

Year: 2024

Director: Test director1

Type: Movie

Cast: 

- Actor 1
- Actor 2

Movie ID: 585a737354556d461e7a166e

[← Back to Movies](#)

[Edit Movie](#)

[Delete Movie](#)

# Assignment 02 - Screenshots /6

## Edit page

[HOME](#) > [SIGN IN](#) > [SIGN UP](#)

### Edit Movie

Editing: Batman Begins

Title:

Year:

Director:

Type:

IMDB Rating:

Poster URL:

Cast:

## Notification messages - Error

### Movies

Error: Invalid-movie-ID-format

## Notification messages - Success

### Movies

Success: ok

# Assignment 02 - Things that could be improved

UI: Visuals, Animations, notifications

Validation: On forms, movie poster URLs, etc

Security: Try to reduce possible XSS, injection attacks

Architecture: Cleaner code