Faculty of Science

| | |
|---|---|
| **Course**: | CSCI 3230u: Web Application Development |
| **Semester:** | Fall 2016 |
| **Course component:** | Final Exam - lab exam |
| **Weight:** | 25% of your final grade |
| **Note:** | There is also a quiz worth 5% of your final grade |

# Instructions

This lab exam is to be completed and submitted to Blackboard before the lecture period has finished. You are permitted any notes or code that you want, as long as they are found on the hard disk of your laptop. Any communication with a live person (aside from the invigilators) is prohibited, as is the use of a file sharing tool (e.g. Google Drive), posting to websites (e.g. forums like StackOverflow), etc.

## Overview

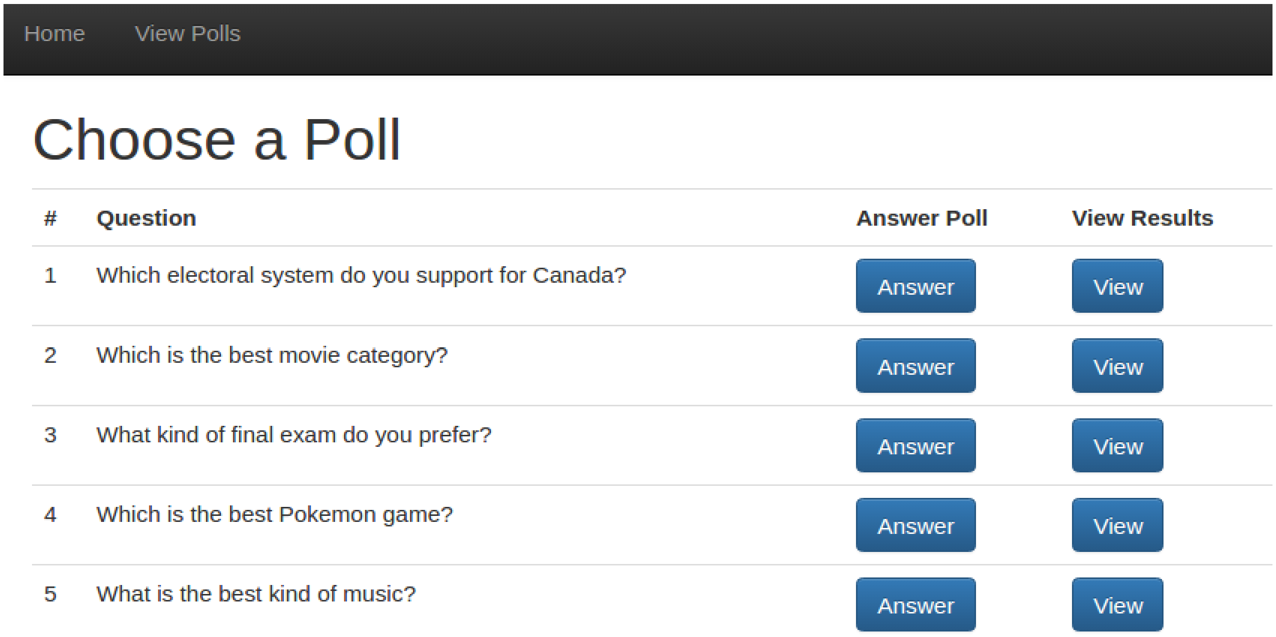
This lab exam involves the modification of an existing web application (`final_exam_base_code.zip`, which can be found in the `Assessments` section on Blackboard). The web application currently displays a table of polls, and for each there are two buttons. The first, `Answer`, links to a page where the user can answer a poll. The second, `View`, links to a page where the user can view the results of a poll. The pages that these two buttons link to, and the other pages that are used by `Answer` and `View`, are your responsibility.

## Detailed Requirements

The base *Node.js* project, along with its `package.json` file, has been provided in the `final_exam_base_code.zip` file. Extract this file to your lab exam working directory. Install all the dependencies, using `npm install`. You will not need any additional dependencies for this project, nor will you need to edit the `package.json` file.

### *Styling*

The application already has a basic CSS file, `main.css`, and a `layout.pug` *Pug* template, which together define the look and feel of each page. The `layout.pug` file defines the menu, and imports *Bootstrap*. You should not need to modify either of these files.



Choose a Poll

| # | Question | Answer Poll | View Results |
|---|---|---|---|
| 1 | Which electoral system do you support for Canada? | Answer | View |
| 2 | Which is the best movie category? | Answer | View |
| 3 | What kind of final exam do you prefer? | Answer | View |
| 4 | Which is the best Pokemon game? | Answer | View |
| 5 | What is the best kind of music? | Answer | View |

**Figure 1 – The output from the (existing) `/listPolls` route**

## *Initial Data*

The data used in the above output comes from a Mongo database. The initial data for this database is provided below, as a series of commands to make it easier for you. This data has also been included in the file `starting_data.txt`. Insert this data into a new database, called `pollit` in your own *MongoDB* server.

*Note: Try not to copy/paste code from this document into your program. Use the text file version, as PDF documents sometimes have unprintable characters when you copy/paste code from them.*

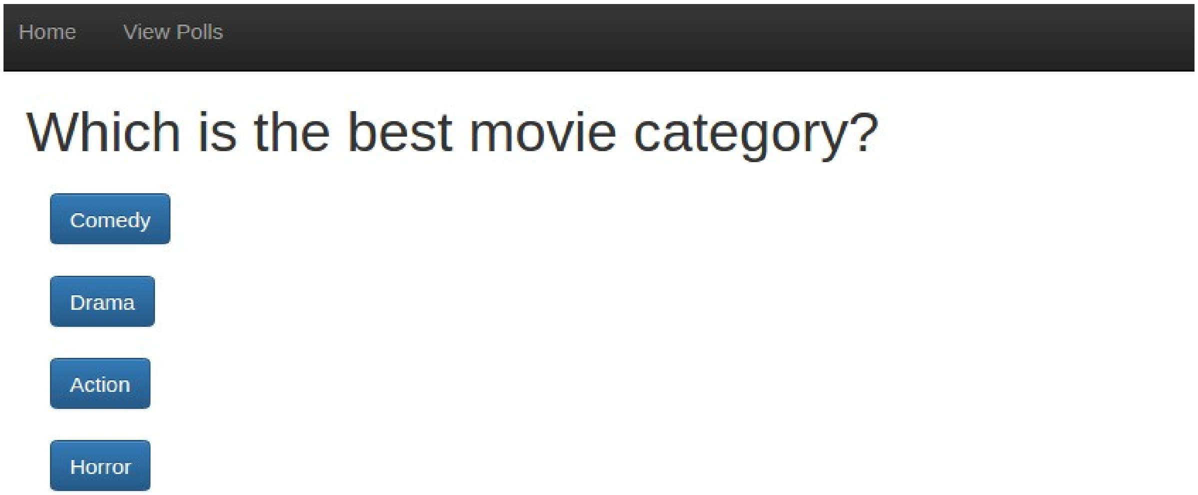
### Code Listing 1 – Initial Data for MongoDB

```
db.pollQuestions.insert({
  questionNum: 1,
  question: 'Which electoral system do you support for Canada?',
  answers: [
    {answerNum: 1, answer:  'Winner takes all'},
    {answerNum: 2, answer:  'Proportional'},
    {answerNum: 3, answer:  'Ranked'}
  ]});


db.pollQuestions.insert({
  questionNum: 2,
  question: 'Which is the best movie category?',
  answers: [
    {answerNum: 1, answer:  'Comedy'},
    {answerNum: 2, answer:  'Drama'},
    {answerNum: 3, answer:  'Action'},
    {answerNum: 4, answer:  'Horror'}
  ]});


db.pollQuestions.insert({
  questionNum: 3,
  question: 'What kind of final exam do you prefer?',
  answers: [
    {answerNum: 1, answer:  'Easy, multiple choice'},
    {answerNum: 2, answer:  'Tough, short answer'},
    {answerNum: 3, answer:  'Easy, lab exam'},
    {answerNum: 4, answer:  'Tough, lab exam'},
    {answerNum: 5, answer:  'Tough, essay'}
  ]});


db.pollQuestions.insert({
  questionNum: 4,
  question: 'Which is the best Pokemon game?',
  answers: [
    {answerNum: 1, answer:  'Pokemon Red'},
    {answerNum: 2, answer:  'Pokemon Yellow'},
    {answerNum: 3, answer:  'Pokemon Emerald'},
    {answerNum: 4, answer:  'Pokemon Black'},
    {answerNum: 5, answer:  'Pokemon Alpha Sapphire'},
    {answerNum: 6, answer:  'Pokemon Moon'}
  ]});


db.pollQuestions.insert({
  questionNum: 5,
  question: 'What is the best kind of music?',
  answers: [
    {answerNum: 1, answer:  'Pop'},
    {answerNum: 2, answer:  'Rock'},
    {answerNum: 3, answer:  'Hip hop'},
    {answerNum: 4, answer:  'Metal'},
    {answerNum: 5, answer:  'Jazz'},
    {answerNum: 6, answer:  'Country'}
  ]});
```

### *Part 1: Answering Polls (10 marks)*

For this part of the application, you will need to perform the following steps:

1. (3 marks) Write a `GET` handler route (in Node.js) for the URI `/answerPollQuestion`
   - This route will collect the question number from the provided parameter
   - The route will then load the question and answer data already in the MongoDB
     - There is already a Mongoose schema for this data in `app.js`
   - The data will be passed to `answer_poll.pug` (from the next step) for display
2. (2 marks) Develop a view, called `answer_poll.pug`, that uses Pug to display the question text and show each of the answers as clickable buttons
   - This view will display the question as an `<h1>`, and each of the answers as a button
     - The answer buttons will be displayed vertically
     - A sample question output is given in figure 2, below
     - Each button will submit their answer (using `POST`) to `/processResponse`



3. (2 marks) Create a Mongoose Schema, called `PollResponse`, which matches the data format given below
   - This schema will be used when inserting the user's response into the MongoDB
   - An example of a response (in JSON), is given below:

**Code Listing 2: An example poll response**

```
{
        "questionNum" : 3,
        "answerNum" : 2,
        "_id" : ObjectId("5846e8d206464853f303c89d"),
}
```

4. (3 marks) Write the `POST` handler route for the URI `/processResponse`
   - This route will collect the data submitted by the form from step 2 and insert it into MongoDB using the schema from step 3
   - When finished, redirect the user to the `/pollResults` URI with the correct question number included as a parameter

## *Part 2: Viewing Poll Results (15 marks)*

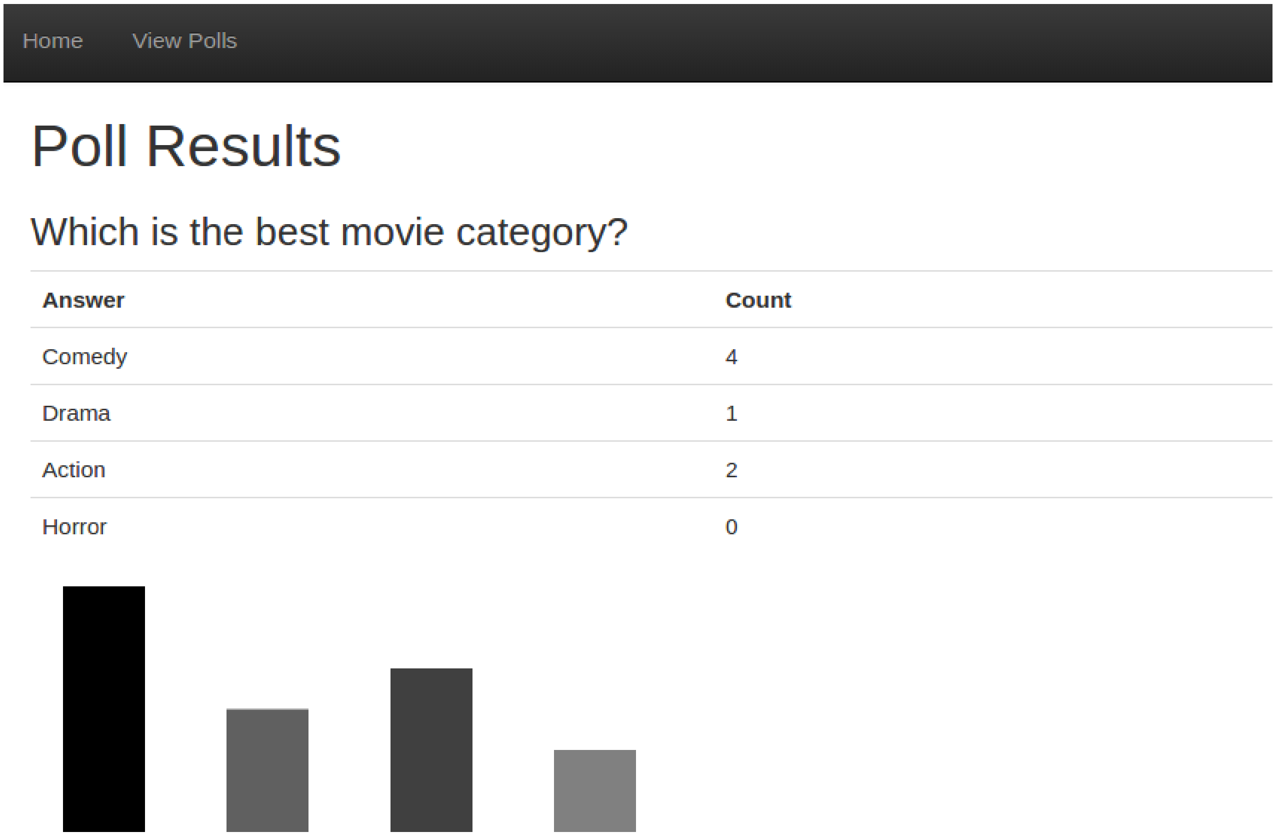
For this part of the application, you will need to perform the following steps:

1. (4 marks) Write a `GET` handler route for the URI `/pollResults`
   - ○ This route will first collect the question number from the provided parameter
   - ○ The route will then load the question/answer and response data for that question from the MongoDB
     - ▪ There is already a Mongoose schema for the question/answer data in `app.js`
     - ▪ A Mongoose schema for the response data was added to `app.js` in the first part
   - ○ The data will be passed to `poll_results.pug` (from the next step) for display
2. (3 marks) Write a `GET` handler route for the URI `/pollResultsJSON`
   - ○ This route will collect the question/answer and response data for the current question, and will output them in JSON format
   - ○ An example of the JSON output that is required is given below:

**Code Listing 3 – Sample Output from `/pollResultsJSON/3`**

```
[{"answer":"Easy, multiple choice","count":11},
 {"answer":"Tough, short answer","count":9},
 {"answer":"Easy, lab exam","count":2},
 {"answer":"Tough, lab exam","count":3},
 {"answer":"Tough, essay","count":2}]
```

3. (2 marks) Develop a view, called `answer_poll.pug`, that uses Pug to display the question text and show each of the answers as clickable buttons
   - ○ This view will display the results of the poll question as an HTML <table>
   - ○ The view will also load a JavaScript file, `poll_results_chart.js`, which is described in the next step
   - ○ A sample response output is given in figure 3, below



Poll Results

Which is the best movie category?

| Answer | Count |
|---|---|
| Comedy | 4 |
| Drama | 1 |
| Action | 2 |
| Horror | 0 |

**Figure 3 – Sample output for /pollResults**

4. (6 marks) Write the `poll_results_chart.js` JavaScript code, which will
   - Submit an AJAX request (using one of D3's AJAX functions) to `/pollResultsJSON` to obtain the response and question/answer data
   - Use D3 to create a (vertical) bar chart of the response data, as shown in figure 3, above

# How to Submit

Collect all of your files (your entire Node.js project, without the `node_modules` directory) and put them into a single ZIP file (e.g. `FinalExam_LabExam_RandyFortier.zip`), which should be submitted to Blackboard, using the drop box for this lab exam.

***Note:*** *Please do not use* `.rar` *or* `.7z` *formats for your submission.*