CIT 253 Programming Project 6

The assignment here is to write an app using a database named CIT321 with a collection named students; we will provide a CSV file of the data. You need to use Vue.js to display 2 pages. You should know that this assignment is similar, all too similar in fact, to the cars4sale example in the lecture notes for Vue.js 2. You should study that program first. If you figure out cars4sale, then program 6 will be extremely straightforward. It is not my intent do drop a ton of new material here in the last few days of class.

The database contains 51 documents. The first rows of the CSV file look like this:

sid	last_name	first_name	major	hrs_attempted	gpa_points
1	Astaire	Humphrey	CIT	10	34
2	Bacall	Katharine	EET	40	128
3	Bergman	Bette	EET	42	97
4	Bogart	Cary	CIT	11	33
5	Brando	James	WEB	59	183
6	Cagney	Marlon	CIT	13	40

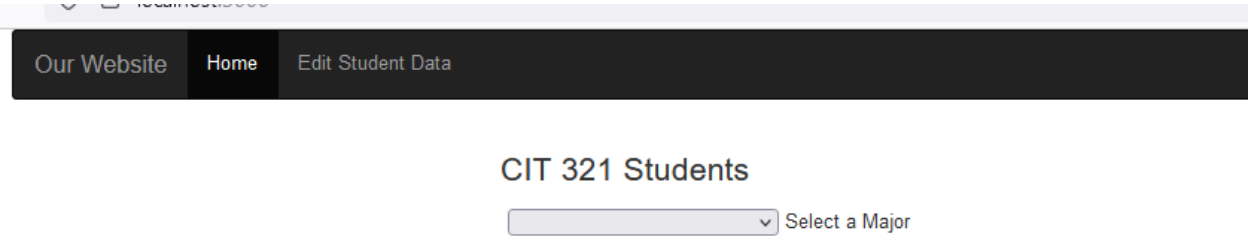
GPA is calculated as gpa_points divided by hrs_attempted. GPA points would have been arrived at by adding 4 points for each credit hour of A, 3 points for each credit hour of B, and so on. Humphrey Astaire would have a gpa of 34/10 = 3.4.

The link to the CSV file will be on the course website. It is named CIT321.csv.

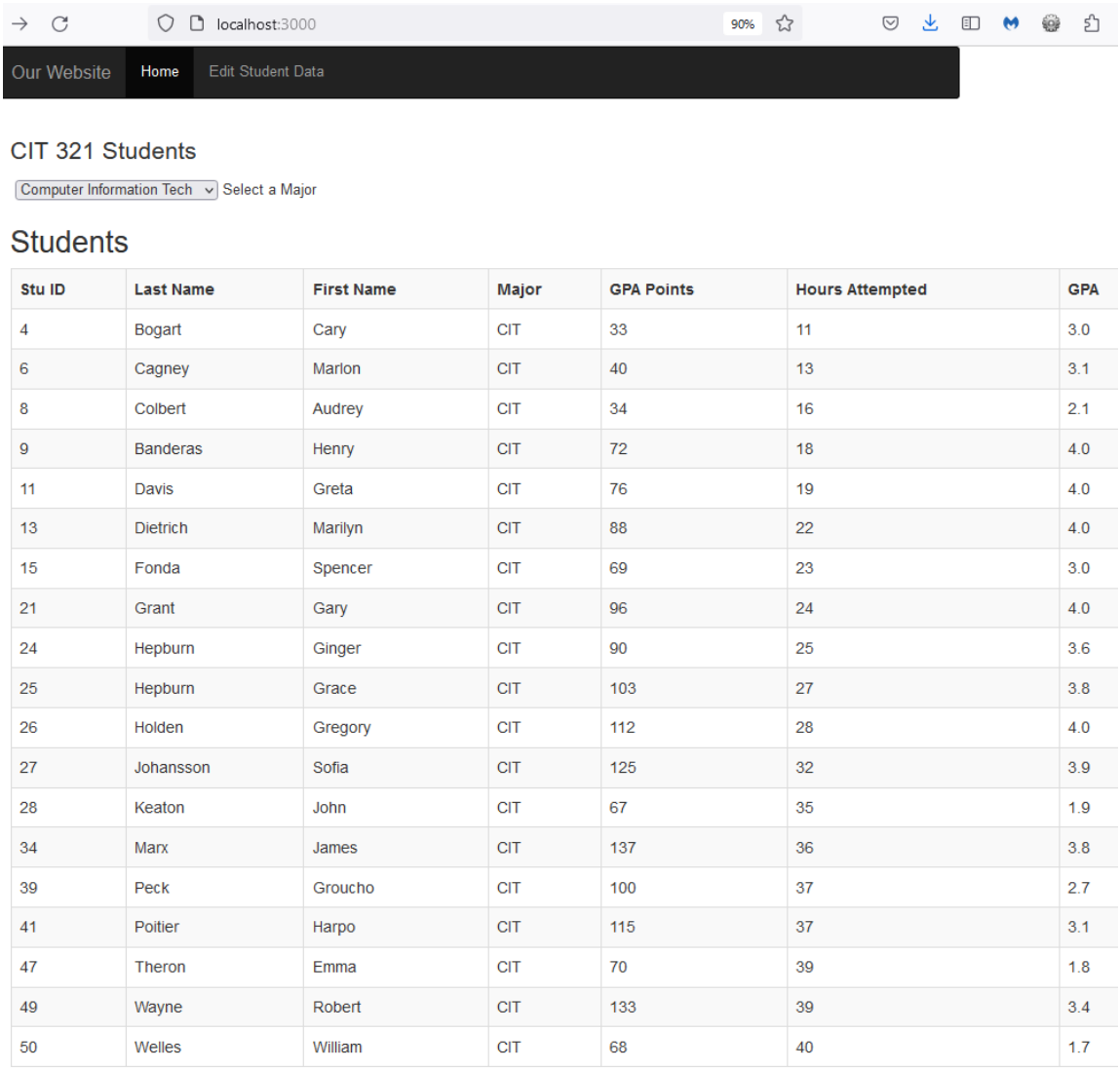
Here is what your model should look like. The model is named Student.js

```
var mongoose = require('mongoose');

mongoose.connect('mongodb://127.0.0.1:27017/CIT321');

var studentSchema = new mongoose.Schema({
     sid: {type: Number, required: true, unique: true},
     last_name: {type: String, required: true},
     first_name: {type: String, required: true},
     major: {type: String, required: true},
     hrs_attempted: {type: Number, required: true},
     gpa_points: {type: Number, required: true}
});

module.exports = mongoose.model('Student', studentSchema);
```

Here's what the pages should look like, more or less. You have 2 pages, index.html and edit.html in the public folder. The navbar has links named *Home* and *Edit Student Data*. Home opens the index.html page that has a drop-down list that will allow us to filter the students by major



If you click the arrow and select CIT, for example, you get something that looks like this.



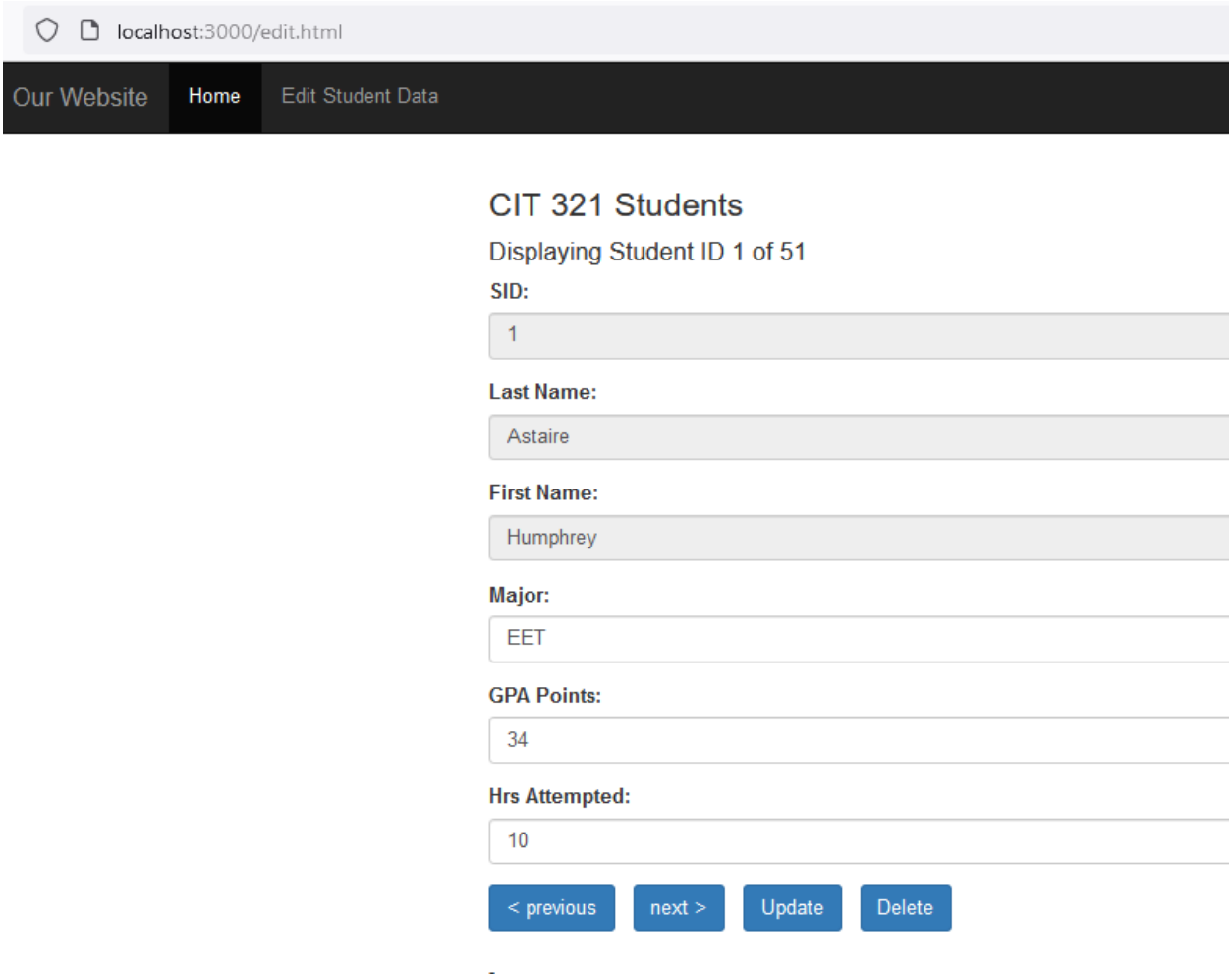
Stu ID	Last Name	First Name	Major	GPA Points	Hours Attempted	GPA
4	Bogart	Cary	CIT	33	11	3.0
6	Cagney	Marlon	CIT	40	13	3.1
8	Colbert	Audrey	CIT	34	16	2.1
9	Banderas	Henry	CIT	72	18	4.0
11	Davis	Greta	CIT	76	19	4.0
13	Dietrich	Marilyn	CIT	88	22	4.0
15	Fonda	Spencer	CIT	69	23	3.0
21	Grant	Gary	CIT	96	24	4.0
24	Hepburn	Ginger	CIT	90	25	3.6
25	Hepburn	Grace	CIT	103	27	3.8
26	Holden	Gregory	CIT	112	28	4.0
27	Johansson	Sofia	CIT	125	32	3.9
28	Keaton	John	CIT	67	35	1.9
34	Marx	James	CIT	137	36	3.8
39	Peck	Groucho	CIT	100	37	2.7
41	Poitier	Harpo	CIT	115	37	3.1
47	Theron	Emma	CIT	70	39	1.8
49	Wayne	Robert	CIT	133	39	3.4
50	Welles	William	CIT	68	40	1.7

Note the GPA is calculated and displayed.

You can calculate the gpa a couple of ways, but this will work.

```
<td>{{(student.gpa_points / student.hrs_attempted).toFixed(1)}}</td>
```

On the other hand, when you click the navbar link for *Edit Student Data*, you get this, with the first record displaying as the page loads.



The operation of this page is described in the lecture notes for the week of Vue.js 2 in the cars4sale example. But The buttons should allow you to step forward and backward thru the data and not exceed the limits, and you should be able to update the major, gpa points and hrs attempted of a student. You should also be able to delete a student.

Here are the requirements.

- Use Vue.js and Bootstrap. Your pages do not need to look exactly like mine, but they need to convey the same information.
- Your database and collection, along with your model, must be compatible with mine.
- The index page should allow filtering all students by major and displaying their GPA

- The Edit Student data pages should allow displaying each document one at a time, allow editing major, gpa points and hrs attempted.
- You should be able to delete a student.
- Oh yeah, and separation of concerns; you need to put your two JavaScript files somewhere in the public folder and link them to the appropriate html page.
- Your routes in index.js should only have to provide data. Formatting is done thru Vue.js

I am still using mongoose 6x, but if you use promises rather than callbacks, then you are welcome to use mongoose 7.

This app is not perfect; just make it work. There is not time enough to luxuriate in the details at this point.