COSI 152A – Web Application Development

Creative Project Assignment 6

Objectives:

- Database Connectivity with MongoDB
- Mongoose Schema and Models
- Controller Actions
- CRUD Functions
- Flash Messages

Task:

Use your CPA5 assignment and build upon it by connecting your application to MongoDB and creating the relevant schema and models. Associate models with each other and create an object literal of all the controller actions in a controller file. Apply CRUD functions to your application by creating the relevant views, controllers actions and routes.

Requirements:

- Your application has to fully apply the MVC articheture by building the relevant Models, Views and Controllers. Sample structure is provided at the end of this document.
- Separate models should have separate views folder and separate controllers.
- Your project has to follow all the latest code modifications and improvements that we have made during the class. For example, use router.get() or router.post() instead of app.get() or app.post() for writing the routes.
- Avoid code redundancy by defining a separate block of code as a function or action once and use it several times when needed.
- Your project should apply the CRUD operations on all the models you create in this project.
- Add clear and explainatory comments for your controller actions, models and schema, and for the routes in the main application file.

Models:

Create the following three models with their relevant schema validators and properties:

- User model:

```
name: { type: String, required: true },
email: { type: String, required: true, unique: true },
password: { type: String, required: true },
role: { type: String, enum: ["student", "alumni"], default: "student" },
graduationYear: { type: Number, required: true },
major: { type: String, required: true },
job: { type: String },
company: { type: String },
city: { type: String },
state: { type: String },
country: { type: String },
zipCode: { type: Number, min: 10000, max: 99999 },
bio: { type: String },
interests: [{ type: String }],
```

- Event model:

```
title: { type: String, required: true },
description: { type: String, required: true },
location: { type: String, required: true },
startDate: { type: Date, required: true },
endDate: { type: Date, required: true },
isOnline: { type: Boolean, default: false },
registrationLink: { type: String },
organizer: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
attendees: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }]
```

- Job model:

```
title: { type: String, required: true },
company: { type: String, required: true },
location: { type: String, required: true },
description: { type: String, required: true },
requirements: { type: String, required: true },
salary: { type: Number, required: true },
contactEmail: { type: String, required: true },
contactPhone: { type: String, required: true },
postDate: { type: Date, default: Date.now },
deadlineDate: { type: Date, required: true },
isActive: { type: Boolean, default: true },
```

Note: Consider the association between event and user models. Event organizer and event attendees are instances of the user model. Based on this association, provide an option at the end of every event if someone is interested to attend. If interested to attend, check if he/she is already a user or not, if not, direct him/her to the corresponding view to create a user account first.

Views:

Create the neccessary views for all the CRUD operations for all the three models in separate folders within views folder.

- You can add more views if required.
- Among many views for the Job and Event models, link the Jobs and Events options of your navigation bar to only the views in which you generally display all the jobs or all the events in your application. in other words, clicking on Jobs or Events options of your navigation bar should take the user the views where all the Jobs or events are displayed.

Controllers:

Create three separate controllers for the three models and add the relevant actions.

- Create controller actions with proper name for all the CRUD operations and for all the three models.
- Avoid creating duplicated code actions in a controller file. Create functions and call several times where applicable.
- Export all actions of a model's controller using a single export statement.
- Redirect to proper views on completion of a CRUD operation. This should be applied for all the three models.
- Create proper flash messages when reloading or redirecting after a potential success or failure operation.
- Call the relevant controller action within the main application file while writing the route for requests.

Assignment will be graded based on:

- Internal Correctness: The assignment should meet all the requirements.

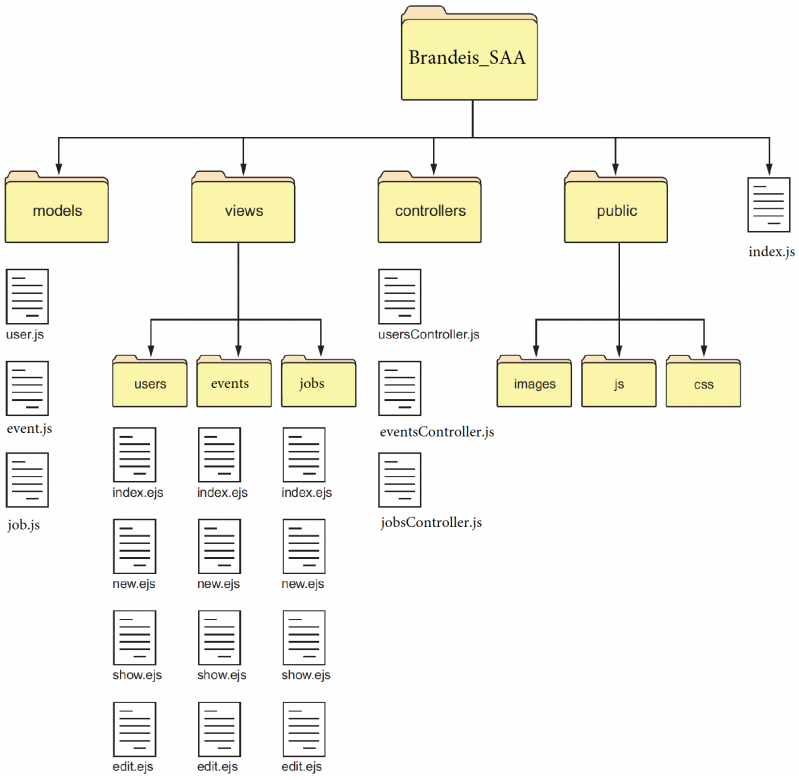
Assignment submission guideline:

1. Your project name is Brandeis_SAA
2. Submit it via Latte in a zip folder named yourFirstname_yourLastname_CPA6.zip

Submission deadline:

Please make sure to submit your assignment before 11/04/2023 11:59 PM

Directory structure sample



Note: this is a sample. Number of files and folders and their names may vary according the requirement.