Web Programming – Spring 2021

Project

Open Date - May 24, 2021 **Total Marks-300**

Deadline -20 June 2021, 11.00pm

Overview

1. Develop a full stack web application with a dynamic front end, with a document-oriented database, and REST API.
2. This project is group task and you are allowed to work in group of maximum 4 students.
3. cross section groups are not allowed.
4. Each group is required to submit a project idea and group members **no longer than 29th May 2021** on below link**.**
   https://forms.gle/GpDpQWvzn5Jm5Y7Q6
5. Plagiarism is strongly forbidden and will be very strongly punished. If we find that you have copied from someone else or someone else has copied from you (with or without your knowledge) both of you will be punished. You will be awarded straight zero.
6. Each group can take help from internet resources but 70% of code should be written by the group and 30% code can be taken from online sources such as GitHub etc. If you are taking some code from online source, please mention it in your submission, otherwise it will be considered as Plagiarism and will be panelized.
7. There will be penalty of 20% marks deduction on Late submissions.

Objective

- Implement web front end using html, css and bootstrap
- Learn to implement client-side frontend using any of JavaScript framework (angular, react, Vue)
- Implement a system according to specifications that are provided by someone else.
- Learn to implement Object oriented analysis and design
- Demonstrate an understanding of server-side concepts, CRUD and REST
- Build and configure a backend server using NodeJS framework
- Build a RESTful API for the front-end to access backend services

Instructions

1. You are required to get an idea of your own and develop a web application based on your choice.
2. You can choose any of UI design and UI JavaScript framework for your web application but it must be user friendly.
3. Landing page of your application should contain interactive dashboard features.

Client Side -Technical Requirements [100]

Below these are the requirements that all submissions need to satisfy:

- Web Application should have at least 4 pages/components. [20]
- Each page should include these components:[20]
  - A relatively large image at the top. The image needs to be automatically changed every 5 seconds.
  - A navigation menu that includes links to all pages in your website. There should be at least 3 links in this section linking to each of the required pages mentioned above.
  - A content section which will be different for each page.
  - A footer section which shows the copyright notice, Contact Details and any other information you would like.
- All pages in your website should have the same layout. Specifically, all components mentioned above should be identical in all pages except for the content section.
- You will use the async/await and the fetch () function to make network requests that allow your client to communicate with your server's API.[20]
- Your client-side code will be served through your express web server.[10]
- Your web Application must include the Routing module [30]

Functional Requirements [200]

Your Web application must have these functionalities implemented

- Signup [20]
- Login [20]
- Two separate Dashboard required, one for admin and one for user.
  - Dashboard – You should display data in tabular form from at least 3 tables/documents using aggregation [30 *2 = 60]
- Search required data from database and display on page [20]
- Pagination [20]
- Image upload [10]
- File upload [10]
- CRUD implementation of at least 4 Entities of your system [40]

Backend Design

- Before figuring out what your API and database look like, it's helpful to start from what you want your website to be capable of. What's the point of working on an endpoint that a user will never interact with?
- Make some sketches, wireframes, visual designs, and/or a style guide.
- Make a list of requirements.
- After making a list of requirements, try to make a plan for how you will accomplish this with code. This will include things like
  - Data Model
  - API endpoints
  - User events
  - JS/HTML/CSS Pseudocode

- **Server-Side**:
  - You will use npm to manage your dependencies
  - You will use express to create your api
  - Your endpoints will be documented
  - Your server-side code to define meaningful API routes that allow you to CREATE, READ, UPDATE, and DELETE data on the server using http requests. You will use:
    - GET: app.get()
    - POST: app.post()
    - PUT: app.put()
    - DELETE: app.delete()
  - You will create multiple endpoints that do at least one of the following:
    - Access an API that can't be used client-side,
    - You will read and write to a JSON file data store
  - Your client side code will be served through your web server by:
    - creating a static file server middleware
    - setting the default API route of your server to send the index.html file in your static file directory.
- **Database**:
  - You can use any of document-oriented database of your choice but we will recommend to use MongoDb.

Just Code it and Be a full stack Developer!

Happy Coding.