Assignment - 4.

In this assignment you need to modify Assignment 3, **(related to your chosen topic**).

Please ensure that you check your work before uploading it to Moodle.

If the students' work fails to operate on teacher's computer, they will get 0. Points will only be awarded if the project meets the specified requirements and is *fully operational*. In case if your works will be similar to your classmates or other students from different groups, you will be awarded with 0 grade. This includes designs as well, so try to implement unique design for your work. Develop your frontend using EJS.

The server is expected to run on port 3000, and the submission should include a package.json file listing all dependencies. **Ensure that you grant IP access to your Atlas from any location**. Additionally, *a navigation bar* must be implemented on all pages, featuring a well-designed layout for seamless redirection.

Provide comprehensive documentation, including setup instructions, API usage details, and explanations of key design decisions in your Readme.md
Add **your name** and **group number** in your footer.

Failure to comply with these fundamental criteria, to provide an aesthetically pleasing design with a functional navigation bar may result in point deductions.
Furthermore, ensure that all necessary API keys, including your own, are included. Failure to provide the required API keys may lead to point deductions. Your project have to be deployed, in case if it is not, it may result in point deductions.

Assignment Requirements and Grading:

Authentication and Authorization: 1. **User Registration**: • Create a registration page where users can sign up by providing a username, password, and any other required information. • When a user registers, hash their password using bcrypt before storing it in the database along with other information like username, creation date, etc. • Ensure that the username is unique to avoid conflicts. 2. **User Login**: • Create a login page where users can input their username and password. • Retrieve the user's hashed password from the database based on the provided username. • Use bcrypt to compare the hashed password stored in the database with the password entered by the user during login. • If the passwords match, authenticate the user and redirect them to the main page. 3. **Authentication Middleware**: • Implement middleware to protect routes that require authentication. • This middleware should check if the user is logged in by verifying the presence of a session token or any other authentication mechanism you're using. • If the user is not authenticated, redirect them to the login page. 4. **Authorization**: • Define user roles or permissions such as 'admin' or 'regular user'.	20%

• Store the user's role in the database. • Implement authorization checks on routes that require specific permissions. • For example, only allow administrators to access the admin page or perform administrative actions. It is essential that your admin username is your name. Additionally, include detailed password information in the README file. **Note:** Execute your logic within the core JavaScript file of the server	
REST API: Implement a functionality within your admin page enabling the addition of new items related to your topic to your main page. Each item should include three pictures, two names for localization in different languages, two descriptions for localization, and timestamps for creation, update, and deletion. Admins should be able to edit, delete, and add these items. On the main page, display these items in well-designed blocks, each featuring a carousel showcasing the three pictures. Ensure that each block also displays the name and description of the item.	35%
APIs: Add two different APIs related to your topic. Implement multi-language support, allowing users to select their preferred language for a personalized experience. **Note**: If your APIs were previously focused on mapping, weather, or geolocation, you'll need to replace them with new ones that offer valuable data and differ from your API choice.	30%
Project Organization and Design **Clean Code and Project Structure:** Keep your code clean, well-documented, and organized. Follow best practices for coding and maintain a clear project structure that you learned from previous assignment. (readme file) **Responsive Design and User Interface:** • Enhance the user interface with thoughtful design elements, making the application visually appealing with EJS.	15%
Overall you can get	100%
Bonus task. Add this to your navbar as “Bonus”. *To clarify, students must fully implement both parts of the bonus task in order to receive any marks for it. Each part is considered essential for the completion of the bonus task, and partial implementation will not be awarded any points.* **1. Timed Quizzes (at least 5 questions):** • Implement a countdown timer for each quiz session, allowing users a limited amount of time to complete the quiz. • Display the remaining time to users and automatically submit the quiz when the time runs out. • Provide feedback to users on whether they completed the quiz within the time limit. **2. Social Sharing:** • Integrate social media sharing buttons (e.g., Facebook, Twitter, LinkedIn) within the quiz interface.	10%

• Enable users to share their quiz results or invite friends to participate in the quiz. • Include dynamic sharing content, such as a summary of quiz performance or an intriguing quiz question, to encourage engagement. **Note: Store quiz questions, answer options, correct answers, and related metadata (e.g., category, difficulty level) in MongoDB Atlas.**	