Biostate.AI Full Stack Developer Coding Challenge

Overview
As part of our hiring process for a Full Stack Developer position at Biostate.ai, we'd like you to complete a coding challenge that demonstrates your skills in both frontend and backend development. This challenge consists of two separate features based on string algorithms and binary tree operations, with a focus on both functionality and user interface design.

Task Description
You are required to create a web application that implements two main features. These features are independent of each other and should be implemented as separate modules within your application.Please submit your completed challenge within two weeks of receiving this task.

Note 1: If you find any part of the task description vague or unclear, feel free to interpret it as you see fit. However, please explain your interpretation in your submission.
Note 2: Please be aware that one of the key evaluation criteria is the time you take to complete this task. This is because we are a startup, and for us, time is one of our most important assets. While we give you two weeks to complete this challenge, if you were to do it in a shorter time, we would see you as a much higher quality candidate than someone else who gives a comparable solution in two weeks.

Feature 1: Longest Substring Calculator
Create a page where users can input a string, and the application will perform the following operations:

a. Find the length of the longest substring without repeating characters.
    i. Example: For input "abcabcbb", the output should be 3 (for "abc").
    ii. Example: For input "bbbbb", the output should be 1 (for "b").
b. Display all substrings that contain only unique characters.
    i. Example: For input "abc", the output should be ["a", "b", "c", "ab", "bc", "abc"].
    ii. For longer strings, limit the output to substrings of length 10 or less to avoid overwhelming the user.

Additional requirements:
- Implement input validation to ensure the string contains only valid characters (e.g., alphanumeric and basic punctuation).
- Provide clear error messages for invalid inputs.
- Display the results in a visually appealing and easy-to-read format.
- Implement a feature to allow users to save their calculations and view their history.

Visual and UI requirements:
- Create an intuitive input form with a clean, modern design.
- Use a color-coded system to highlight the longest substring without repeating characters in the original input.
- Display the list of unique substrings in a scrollable container, with the ability to copy individual substrings.
- Implement a visual indicator (e.g., progress bar or spinner) for calculations on longer strings.
- Create an attractive layout for the calculation history, possibly using cards or a table format.

Feature 2: Binary Tree Path Sum Calculator
Create a page where users can input a binary tree structure, and the application will perform the following operations:

a. Calculate and display the maximum sum path from a leaf node to any node.

        i. Example: For the tree [10, 5, -3, 3, 2, null, 11, 3, -2, null, 1], the output should be 23 (5 -> 2 -> 1).
    b. Calculate and display the maximum sum path between any two nodes.
        i. Example: For the same tree, the output should be 27 (11 -> -3 -> 10 -> 5).

Additional requirements:

- Provide a user-friendly interface for inputting the binary tree structure (e.g., a visual tree builder or a specific input format).
- Validate the input to ensure it represents a valid binary tree.
- Visualize the binary tree structure and highlight the maximum sum paths.
- Allow users to save and load tree structures.

Visual and UI requirements:

- Implement an interactive tree builder where users can add, edit, or delete nodes visually.
- Create an aesthetically pleasing visualization of the binary tree, with clear node values and connections.
- Use animations or transitions when highlighting the maximum sum paths on the tree.
- Implement a zoom and pan feature for larger trees to ensure all nodes are accessible.
- Design an intuitive interface for saving, loading, and managing saved tree structures.

Technical Requirements

1. Frontend:

- Use a modern TypeScript framework (React, Vue, or Angular).
- Create a responsive design that works well on both desktop and mobile devices (minimum supported resolution: 320px width).
- Implement proper form validation and error handling for all user inputs.
- Use a state management solution appropriate for your chosen framework (e.g., Redux for React, Vuex for Vue, or NgRx for Angular).
- Implement client-side routing to navigate between different features and pages.
- Utilize modern CSS techniques (e.g., Flexbox, Grid) for layout design.
- Ensure accessibility (WCAG 2.1 AA) for all interactive elements.

2. Backend:

- Implement a RESTful API using Node.js (.Net(C#) or Nest.js).
- Create separate endpoints for each feature (substring calculations and binary tree operations).
- Implement proper error handling and validation on the server-side.
- Use middleware for common operations like logging and authentication.
- Implement rate limiting to prevent API abuse.

3. Data Storage:

- Use a database of your choice (SQL or NoSQL) to store user inputs, results, and user data.
- Implement basic CRUD operations for managing saved calculations and tree structures.
- Ensure proper data validation and sanitization before storing it in the database(Create primary and foreign key relationships).
- Implement efficient querying for retrieving user history and saved items.

4. Authentication:

- Implement a secure user authentication system (registration and login) using industry-standard practices (e.g., JWT, bcrypt for password hashing).
- Secure all API endpoints to require authentication.
- Implement user roles (e.g., regular user, admin) with appropriate access controls.

5. Testing:
   - Write comprehensive unit tests for your backend logic, covering edge cases for both features.
   - Implement integration tests for your API endpoints.
   - Write frontend unit tests for critical components and functions.
   - Implement at least one end-to-end test for each main feature using a tool like Cypress or Selenium.
   - Include visual regression tests to ensure UI consistency.

6. Documentation:
   - Provide clear documentation on how to set up and run your project, including any environment variables or configuration needed.
   - Include API documentation using a tool like Swagger or OpenAPI.
   - Document any algorithms used, especially for the substring and binary tree operations.
   - Include inline code comments explaining complex logic or non-obvious decisions.

7. Performance:
   - Implement proper error handling and loading states in the frontend.
   - Optimize database queries for efficiency.
   - Implement caching where appropriate (e.g., caching frequent calculations or database results).
   - Ensure quick load times and smooth interactions, especially for animations and transitions.

8. Security:
   - Implement CSRF protection.
   - Ensure proper CORS configuration.
   - Protect against common web vulnerabilities (e.g., XSS, SQL injection).

Bonus Points

- Implement real-time updates using WebSockets for collaborative features (e.g., shared workspaces for calculations).
- Add more advanced visualizations for the binary tree and the maximum sum paths (e.g., interactive, zoomable trees).
- Implement a worker thread or background job system for handling long-running calculations.
- Add a feature to compare the efficiency of different algorithms for the substring problem, with performance metrics and visualizations.
- Implement a simple admin panel for managing users and viewing application statistics.
- Create a dark mode option with a smooth transition between light and dark themes.
- Implement keyboard shortcuts for common actions to improve accessibility and power-user experience.

Submission Instructions

1. Code Repository:
   a. Create a public GitHub repository for your project.
   b. Ensure your repository includes a comprehensive README.md with setup instructions.
2. Hosting:

    a. Deploy your application to a free hosting platform of your choice (e.g., Vercel, Netlify, or Heroku).
    b. Ensure that both the frontend and backend are properly deployed and accessible.
3. Submission:
    a. Please send the following information to career@biostate.ai:
        i. Your full name and contact information
        ii. GitHub repository URL
        iii. Deployed application URL
        iv. A brief description of your implementation and any challenges you faced
        v. Any additional notes or features you'd like to highlight
        vi. Send an email to career@biostate.ai with the subject line "Full Stack Developer Coding Challenge Submission - [Your Name]"
4. Interpretation and Additional Features:
    a. If you find any part of the task description vague or unclear, feel free to interpret it as you see fit. However, please explain your interpretation in your submission.
    b. While the description provided outlines the minimum requirements, you are encouraged to add features you believe would enhance the application or demonstrate your capabilities further.
    c. If you choose to add extra features, please clearly describe them in your submission, explaining why you believe they are valuable additions.
5. Deadline:
    a. Please submit your completed challenge within two weeks of receiving this task.

Evaluation Criteria
We will evaluate your submission based on the following criteria:

1. Functionality: Does the application work as described?
2. Time to solution: How long you took to create the solution. We are a startup and for US time to a good, functioning, solution is extremely important.
3. Code Quality: Is the code well-structured, readable, and maintainable?
4. User Experience: Is the interface intuitive, responsive, and visually appealing?
5. Testing: Are there sufficient tests to ensure reliability?
6. Documentation: Is the project well-documented for both users and developers?
7. Problem-Solving: How did you approach the challenges?
8. Creativity: Did you add any interesting or innovative features?
9. Interpretation: If you interpreted any requirements, how well did you explain and justify your choices?
10. Additional Features: If you added extra features, how relevant and well-implemented are they?
11. Visual Design: How well does the UI/UX align with modern design principles and the specified requirements?

Good luck! We look forward to reviewing your submission. If you have any questions, please don't hesitate to reach out to career@biostate.ai .

Remember, Biostate.ai values creativity and initiative. While we've provided a framework for this challenge, we're excited to see how you might expand upon it to showcase your unique skills and vision for the project.