STK - Technical Test

Fullstack Menu Tree System

Overview - The Goals 🎯

Build a fullstack application that implements a hierarchical menu tree system with CRUD operations. The system should support nested menu items with unlimited depth, and implement the UI as similar as possible like in this figma
https://www.figma.com/design/Pa1vc4POl4DGsvDTutwM6k/Fullstack---ReactJS%2FVueJS

📋 Requirements

📋 Backend Requirements

Choose your stack:

- **Option A: NestJS (TypeScript/JavaScript)**
- **Option B: Go (Golang)**

Core Features:

1. RESTful API endpoints for menu management (Create, Read, Update, Delete)
2. Support for hierarchical data structure with parent-child relationships
3. Endpoints for reordering menu items within the same level (if implementing bonus)
4. Endpoints for moving menu items between different parent nodes (if implementing bonus)
5. Input validation and error handling
6. Database integration (PostgreSQL or MySQL)
7. API documentation (Swagger/OpenAPI)

📋 Frontend Requirements

Tech Stack:

- React.js or Next.js
- TypeScript
- State management (Redux, Zustand, or Context API) - (if implementing bonus)
- Tailwind CSS

Core Features:

1. Display menu tree in a hierarchical structure

2. Add new menu items at any level
3. Edit existing menu items inline or via modal
4. Delete menu items with confirmation
5. Drag-and-drop functionality for reordering and restructuring (if implementing bonus)
6. Expand/collapse nested menu items
7. Search/filter functionality
8. Responsive design for mobile and desktop
9. Loading states and error handling

💪 Docker Requirements (Bonus)

- **Development mode:** Docker Compose setup with hot-reloading for both frontend and backend
- **Production mode:** Optimized multi-stage Docker builds
- Include `docker-compose.yml` for easy setup
- Include `Dockerfile` for both frontend and backend
- Database container configuration
- Environment variable management
- Volume mounting for persistent data

Technical Specifications

API Endpoints

GET	/api/menus	- Get all menu items (tree structure)
GET	/api/menus/:id	- Get single menu item
POST	/api/menus	- Create new menu item
PUT	/api/menus/:id	- Update menu item
DELETE	/api/menus/:id	- Delete menu item (and children)
PATCH	/api/menus/:id/move	- Move menu item to different parent (if implementing bonus)
PATCH	/api/menus/:id/reorder	- Reorder menu item within same level (if implementing bonus)

📋 Deliverables

1. **Follow best practices** (service layer, validation, error handling).
2. Complete source code with clear folder structure
3. README.md with:
    - Setup instructions
    - How to run in development mode
    - How to run in production mode
    - How to run with Docker (if implementing bonus)
    - API documentation or link to Swagger
    - Technology choices and architecture decisions
4. Database schema/migration files

5. Environment variable templates (`.env.example`)
6. Docker configuration files (if implementing bonus)
7. Basic test coverage (unit tests preferred) - (if implementing bonus)

✅ Evaluation Criteria

1. **Code Quality** (30%)
   - Clean, readable, and maintainable code
   - Proper error handling
   - Code organization and architecture
   - TypeScript usage (if applicable)
2. **Functionality** (30%)
   - All required features working correctly
   - Proper handling of edge cases
   - Data integrity and validation
3. **User Experience** (20%)
   - Intuitive interface
   - Responsive design
   - Loading and error states
4. **Documentation** (10%)
   - Clear setup instructions
   - API documentation
   - Code comments where necessary
5. **Bonus Points** (10%)
   - Docker implementation
   - Test coverage
   - Performance optimizations
   - Smooth drag-and-drop interactions

🧭 Submission Guidelines

1. Push code to a GitHub/GitLab repository (public)
2. Include all necessary files to run the application
3. Provide a detailed README with setup instructions
4. Include screenshots or video demo
5. Share the repository link

⏰ Duration

Finish within 3 days

Good luck with the test! 🚀