Aircaft maintenance task visualization

Programming Challenge

Revision: 2024-04-23

Background

An airline is currently facing challenges to present aircraft maintenance tasks in the timeline, as it currently relies on manual work via Excel. Your task is to help the airline visualize the aircraft maintenance tasks in the timeline.

Task

Build a full-stack application using Node.js with TypeScript for the backend and React.js (Vite) with TypeScript for the frontend. The application should implement a timeline view that displays data on flights and maintenance tasks(work packages). Additionally, the backend should provide a RESTful API for GET operations on these resources, with proper error handling. Sample data will be provided in JSON format as an attachment.

Backend Requirements:

- Choose either an in-memory database solution (e.g., SQLite, NeDB) or Dockerized PostgreSQL for data persistence.
- Implement a RESTful API using Node.js with TypeScript using some modern web framework.
- Define endpoints for GET operation on resources: flights and work packages.
- Ensure good project structure and robust error handling throughout the API.

Frontend Requirements:

- Use React.js (Vite) with TypeScript
- Implement a timeline component that visually represents entities fetched from the backend API in the timeline. The

timeline should render dynamically hourly/daily time range at the top of the component.
  - Flight items should show flight numbers, departure and arrival scheduled times, and station names.
  - Work package items should be put inside corresponding inbound/outbound flights to/from station "HEL" i.e. between inbound and outbound flight arrival and departure times. The work package has information about the flight registration number. Use this and flight information to map work packages between inbound/outbound flights.
  - Work package item should show at least the name, expected start/end times, number of work orders, and status of a work package.
- Display a list of aircraft registrations on the left side of the timeline and render aircraft flights/work packages on the timeline using distinguished "blocks" for flight and work package items. E.g

```
    ABC: |----| |-LHR---FLIGHT 1---HEL-| |----WORK PACKAGE 1---|
|-HEL---FLIGHT 2---LHR-|
```

- Note that there can also be some work packages overlapping each other in time.
- Design a clean and intuitive UI layout with proper styling.
- Implement features like scrolling, dragging, or zooming (optional) for navigating through the timeline.

Additional Instructions:

- Containerize both the backend and frontend applications using Docker.
- Provide clear documentation for setting up and running the applications in the README.md file.
- Write unit tests for critical parts of the codebase.
- Ensure code quality, readability, and maintainability throughout the project.
- Use only well-maintained packages with open-source licenses.

Evaluation Criteria

- Functionality: Does the application meet the specified requirements and perform CRUD operations accurately?
- Error Handling: How effectively are errors handled throughout the application?

- Code Quality: Is the code well-structured, readable, and maintainable?
- Database Integration: How well is the chosen database solution integrated with the backend API?
- UI/UX Design: Evaluate the UI layout, design, and user experience of the frontend application.
- Containerization: Are Dockerfiles and Docker Compose configurations correctly implemented?
- Testing: Are your code features testable?

In coding pay attention to defined evaluation criterias. Also remember it’s not important get everything right. It’s important that you try.

Enjoy problem solving!