**Quick Local Setup Guide –**

1. Create a database ‘Vypex’ and the run the following script into the query edit. 

Or one can find this script here - [Vypex.Backend/vypex-db-script.sql at main · smtanwar1981/Vypex.Backend](https://github.com/smtanwar1981/Vypex.Backend/blob/main/vypex-db-script.sql)

1. Clone Vypex Backend from here - [smtanwar1981/Vypex.Backend](https://github.com/smtanwar1981/Vypex.Backend)

Clone Vypex Frontend from here - [smtanwar1981/Vypex.Frontend](https://github.com/smtanwar1981/Vypex.Frontend)

Needless to mention, install npm packages install nuget packges etc.

1. Run the Vypex backend project. I have set up the service on port – 7196, one is free to update it and make sure to update the same port number in environment.ts file of frontend project.

A screenshot of a computer program

AI-generated content may be incorrect.

Swagger UI should appear in your browser if everything goes well like this –

A screenshot of a computer

AI-generated content may be incorrect.

1. Now run the frontend project and it should look like this –

A screenshot of a computer

AI-generated content may be incorrect.

(Please forgive me for not making it look appealing or good, as I didn’t focus on the UI look but functionalities and features).

**Backend**

Lets me give you a quick overview on the backend architecture. I didn’t use what was there initially in the repository but rather created my own projects according to the requirements mentioned on github.

1.

A screenshot of a computer program

AI-generated content may be incorrect.

**Vypex.Employee.WebApi** – should be the startup project. Holds the leave management api endpoints. In here –

1. I have added one middleware class to log the incoming requests and outgoing responses.
2. A Service extension class to keep the Program.cs file clean.
3. A base controller to control the format of outgoing responses even in case of exceptions.
4. Two controller classes for Employee and Employee Leave management.

**Vypex.Employee.Common** – a simple c# project to keep the entity models and DTOs in one common place along with some more stuff.

A screenshot of a computer program

AI-generated content may be incorrect.

1. VypexServiceResult class in Core folder is the one that wrap up the final response of endpoints.
2. Rest of the classes are self explanatory.

**Vypex.Employee.Interfaces** – just to hold all interface in one single place.

A screenshot of a computer

AI-generated content may be incorrect.

**Vypex.Employee.Repositories** – I have used repository pattern for read/write operation in the database along with Entity Framework because I think for such a small coding challenge it should be enough wherein one needs to just perform some CRUD operations.

A screenshot of a computer

AI-generated content may be incorrect.

**Vypex.Employee.Services –** defines services so as to apply any sort of logic before or after talking to the database.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Frontend**

As mentioned on github page, I tried using Ant-Design library as much as I can and turned out that its been quite helpful. I have made much changes in the frontend applications its just –

1. Moved the styles sheet files to ‘assets’ folder.

A screen shot of a computer

AI-generated content may be incorrect.

1. Added Environment folder that holds the environment settings for now its just the api base URL. One can add different environment.ts files as environment.development.ts, environment.staging.ts etc.

A screen shot of a computer

AI-generated content may be incorrect.

1. Added an Vype Http Interceptor class so as to log the outgoing requests and their respective responses along with error handling and showing user intuitive error message on the screen using Ant-Design library message service.

A screenshot of a computer

AI-generated content may be incorrect.

1. Created a models folder on root level that holds those DTOs and request-response model classes.

A screenshot of a computer

AI-generated content may be incorrect.

1. Added on more component having functionality to show/add/edit leaves.

A computer screen shot of a computer program

AI-generated content may be incorrect.

**Workflow**

1. As can be seen in the home page, I tried making everything simple on the UI. One can see the leaves in number of days and the instances of leaves per Employee.

A screenshot of a computer

AI-generated content may be incorrect.

There is ‘+’ button on the extreme right to see the leave details of an employee. If one ‘click’ on it, leave details should appear like this -   
A screenshot of a computer

AI-generated content may be incorrect.

I kept the leaves details table UI little smaller so that user can easily visualize the difference of the parent (employee table) and child table (leave details).

Edit Leave

Hitting the ‘Edit’ button will transform the current row in editable model where user can update the leave durations as show below -   
A white background with black and red text

AI-generated content may be incorrect.

Needless to say, once done setting the date one can either save it or cancel it.

On successful editing one should see leave update confirmation popup (again Ant-Design) like below -   
A screenshot of a computer

AI-generated content may be incorrect.

And if someone hit the cancel it will ask this –

A screenshot of a computer

AI-generated content may be incorrect.

Delete Leave

There is nothing much to tell about deleting a leave. Delete will simply delete the leave and update the parent table’s Number of days and Leaves Count.

A leave confirmation popup will be there once the leave deleted in the database.

A screenshot of a computer

AI-generated content may be incorrect.

Add Leave

Hitting on Add leave button will simply add a row to the existing leave details table (if there is any). And show user option to add leave duration as show below –

A screenshot of a calendar

AI-generated content may be incorrect.

Please note, since it’s a date range (appropriate) control (Ant-Design) I didn’t need to compare these startDate and endDate validity on the UI. However the validations of these dates are there on the backend.

Once user is done selecting the date range and hit ‘save’, they should be able to see the confirmation message on the top.

A screenshot of a computer

AI-generated content may be incorrect.

If one tries to save empty date range, they should be able to see the validation message like below -   
A screenshot of a computer

AI-generated content may be incorrect.

Leave Overlapping Validation

The validation rule to check the startDate and endDate if they are not overlapping with any existing leave is there in the backend and the UI will also be able to show this error like below , if someone tries to save so –

A screenshot of a computer

AI-generated content may be incorrect.

As can be seen the first leave entry is overlapping with the new one so backend throws a validation error here.

Lets talk about the Requirements mentioned in the github –

The portal requires the following features:

* Displays the list of employees. - Done
* Can add and remove leave days. - Done
* Shows the total leave days taken by an employee on the employee list. - Done
* Leave days must not overlap. - Done
* Leave days can be edited. - Done
* Leave days have a start and end date. No need to track leave hours. - Done
* Filter employees by name on the employee list. – Done

**What we want to see**

**Backend**

* Connect the Employees endpoint to the "real" database. - Done
* Add leave days to the employee model and database. - Done
* Implement the leave days api in the backend. - Done

**Frontend**

**Improve employees list component**

* Add a refresh/reload button to refetch the list of employees. – Not required.
* Use new Angular 19 resources. - Done
* Handle potential API errors. - Done
* Implement search by employee name functionality – Not completed due to time constrains.
  + Minimise the number of requests to the API where possible
* Feel free to change layout and structure of the page. – Not needed

**Implement Edit employee functionality**

* Use Angular forms and Ant Design components for user input. - Done
* Implement employee leave form control as a separate and re-usable component. – Used Smart-Dumb component rather.
  + The user can dynamically add/modify multiple leave entries for an employee. - Done
  + Leave days cannot overlap. - Done
  + Both start and end date are required. - Done
  + Validate that startDate must be before endDate. - Done