We have a payment system where a user has an account with a balance and they can manage payment requests. Create an HTTP API with endpoints that satisfy the following user stories. It should accept and return json.

1. As a user I want to create a payment request.

- *Required fields: amount, date*
- *The customer balance must be enough for the amount provided, if not the request is created and closed immediately with a closed comment "Not enough funds"*
- *If the request is successful the payment is considered pending*
- *Validation errors are shown for missing fields or wrong values*
- *I can see the new payment request in the list of payments marked as "Pending" (see user story no 4)*

2. As a user I want to cancel a payment request

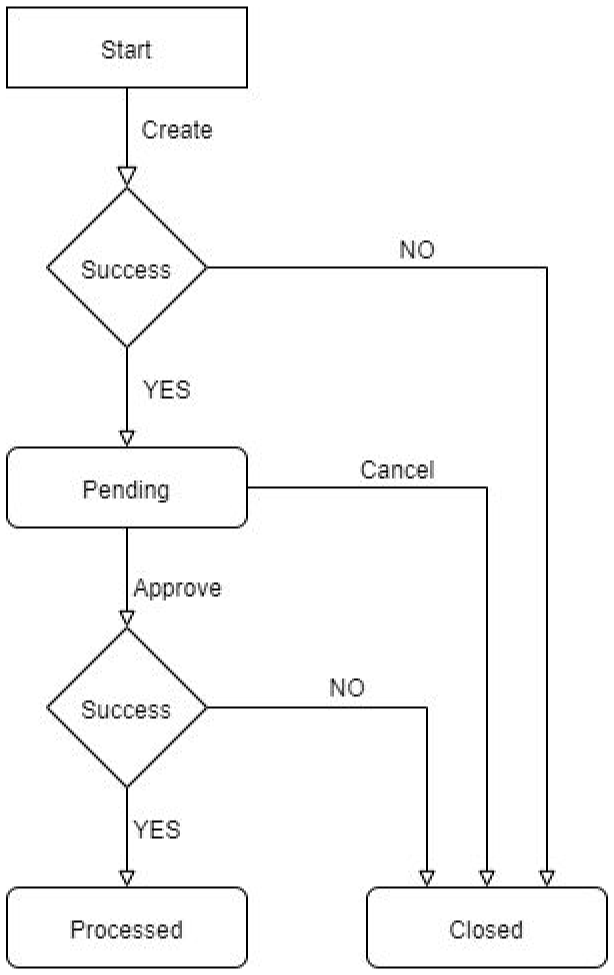
- *Fields (not required): reason (text)*
- *A payment that is processed can't be cancelled*
- *The payment is considered "Closed" with the above closed reason*

3. As a user I want to process a payment request

- *A payment that is closed can't be processed*
- *The payment is considered "Processed"*
- *The balance is reduced by the payment amount*

4. As a user I want to see my balance and a list of payments.

- *Fields: account balance and for the payment list: date, amount, status*
- *Closed reason if it exists*
- *Sorted by newest date*



Notes (please read carefully as they will be considered on top of your implementation of the above):

- Upload your code in an online public repo of your choice and provide us with a link along with instructions on how to build and run in the README.
- We prefer .NET Core but you can select some other framework if you want to.
- We're interested in code architecture and structure, feel free to provide a brief description in your README of the choices you made.
- Assume that this is an enterprise level application that will be worked on by many people and will keep growing. We're looking for extensibility, readability and clean, non-repeated code. **It is important** to add a few comments in your README to describe how you thought about these details and what assumptions you made about the future since it'll be hard to showcase everything in such a small codebase.
- Use any data store that you want, either in-memory data that resets every run, a local file, local SQL storage or cloud storage (please ensure your code is easily configurable and runnable so we can assess it).

- We expect to see properly tested code with good coverage (e.g. unit testing, e2e/functional testing).
- Assume that this application will be deployed through CI to both a UAT and Live environments. Your setup should accommodate for different configurations. Bonus points if you provide CI configuration and/or deploy scripts (comments about how you would deploy are equally acceptable).