

# Senior Software Engineer - (Full-Stack) Technical Assignment

Overview

You are tasked with building a simple web application to view and manage bank accounts. The application should consist of a back end and a front end, demonstrating your ability to handle both server-side and client-side development.

In lieu of a live coding interview, which we understand can be stressful and doesn’t always showcase someone’s coding skills accurately, we decided to give a take home assignment for the technical interview. This assignment gives us the ability to see your coding skills. It is not meant to take more than a few hours, so do as much you can within a reasonable amount of time. Feel free to get creative and use your own designs or add features, but this is not required.

When you are finished with the assignment, upload it to your GitHub or zip your local repository and send it back to the recruiter. Please provide a README that gives instructions for how we can run your code.

Back End

Implement an API based on the attached API design document. This API should implement the endpoints specified in the document.

There are only GET and PATCH operations, which means there is no way to create new data. This was done to keep the task smaller. Because of this, you will have to populate your local database with the provided sample accounts.

1. Framework & Language: Use Node.js with Express or NestJs. We primarily use NestJS, but Express is ok too.
2. No authentication is required on the endpoints. Assume that any client can make unlimited calls to the endpoints.
3. Populate your local DB with the data supplied in `sample-account-1.json` and `sample-account-2.json`
   1. Sqlite or MySQL are preferred DB engines. Sqlite is very easy to run locally, so we recommend that.
      1. If you use MySQL, please provide SQL scripts that we can use to create the schema on our side.
   2. ORMs are optional, but we use TypeORM so that would be a bonus point.
4. API Design is provided in the attached accounts-openapi.yaml.
   1. copy-paste this into editor.swagger.io for an easier viewing experience
5. Error handling:
   1. A 400 should be returned if the request violates the constraints. For example, if the PATCH request has a `name` with a length of less than 1 or greater than 128. Another example is if the `?unmasked` query parameter is not a Boolean value.
   2. A 404 should be returned if the given `accountId` does not exist in the mocks or DB.
   3. The user should not be able to update anything other than the `name` of the account. Other included properties should be ignored and have no effect on the resource.

Front End

Create a front-end application that contains routing, services, features, observables, strong-typing, and more than one module (lazy-loading). This can be as complex or as simple as you want. This is an opportunity to showcase your UI skills, your Angular skills, and an example of your coding best practices.

1. Framework & Language: Angular
2. The nav menu should contain an `Overview` and an `Accounts` drop down, where clicking any account in the drop down will take you to the account details page.
3. The `overview` page should display the accounts, the masked account number, and the available balance
   1. Don’t worry about accounting for other currencies. Assume all balances will be in USD
4. The account details page should display the account type, nickname, opened at date, account number, and routing number
   1. The account number should be masked. Clicking the eye favicon should unmask the account number. This doesn’t have to be a particular icon or favicon. The font-awesome and favicon npm libraries have some that you can use.
   2. The `change` button should provide a way to update the account nickname, and that change should be reflected on the account overview page too.

Sample Front End Design:

* The color schemes are just examples. You are free to use any colors that you choose.
* Don’t worry about the design too much. We have graphic designers at the company who provide mocks, so we don’t expect developers to be graphic designers too.

*Overview Page*

A screen shot of a computer

Description automatically generated

*Account Details Page*

A screenshot of a phone

Description automatically generated

*Updating the Account Nickname*

*A screenshot of a computer

Description automatically generated*

*Unmasking the Account Number*

