1. **Component architecture**

The current draft version of the app has three components (App.component, invoice.component and invoice-dialog.component). The app.compoennt is for rendering the whole application and it uses <router-outlet>

The invoice component shows the currently available invoices and invoice-dialog component is used to add new invoices and will be used to edit current invoices (not implemented yet).

**What went wrong** : due to time concern, I wasn’t able to create a proper structure for the project.

**If I do it all over again** : I would make very small components for instance, the search IBAN will itself will be a component providing semantic web search technique and could be used anywhere in this or other parts of the application.

**Future steps** : I will try to make as many components from the current code as possible.

1. **State management**

The current draft version has the state management using angular basic service and model. Data is loaded and return from the model using services.

**What went wrong:** I thought using angular-redux local storage will take too much time and restoring the whole data on a every change is not necessary. I overestimated the complexity of state management.

**If I do it all over again:** I would go for redux and make the entire app’s state managed there.

**Future steps:** I will start with HTML5 local Storage and later implement redux.

1. **Responsiveness**

The current version is responsive is most screen sizes, only the mobile version requirements from SRSD (Software requirement specification document) is not implemented.

**What went wrong:** again, the complexity of mobile version was underestimate. The problem is that the application needs two different html pages for showing single data, which can never be allowed.

**If I do it all over again:** I would start with mobile version like this time and would complete till the end before I switch to desktop version.

**Future Steps:** I will find a way to show bootstrap accordion on mobile only version there is a bootstrap option, but I will try to design my own solution. And I would remove all inline-styling and use SCSS instead of CSS.

1. **Testing strategy**

Well the best strategy is always unit test to test each unit individually, and then we can use automated testing to fill and submit the forms for performance, errors, validation and stress testing.

**What went wrong:** Testing wasn’t really my concern as that would out of scope of this app.

**If I do it all over again:** I believe I would keep in mind the time required for testing.

**Future steps:** I would write some unit tests and setup an automated testing engine.

1. **User Experience and Design**

The UI design given in the requirements is very well, because it is simple, uses good color combination, fonts are nice, and it is Web Accessibility is good.

But

* page doesn’t use the provided space very efficiently.
* on the mobile view a user doesn’t have the option to switch to bank transfer mode.

**Future steps:** I would add more content probably display both mockups on the same page.

1. **Application architecture**

I believe the smaller the components the better it would, because this uses a form which has variable states so testing small components individually and then doing an integration test would be easy.