**5.1**)

Another libraries used were Formik and Yup. Formik is a lightweight, easy to use form helper library which is concerned in helping with 3 particular points: getting values in and out of form state, validation and error messages; handling form submission (ref - <https://jaredpalmer.com/formik/>). Another advantage is the integration this library has with Material UI, which is used on this project. This requires the use of another library “formik-material-ui” and allows the use and configuration of some Material UI components when building a form (ref - <https://stackworx.github.io/formik-material-ui/>). The Yup library is a lightweight, client-side JavaScript schema builder for value parsing and validation (ref- <https://www.npmjs.com/package/yup>). On the context of this project it is used for validation when building Formik scripts making the validations less verbose (ref - <https://jaredpalmer.com/formik/docs/tutorial#schema-validation-with-yup>).

**Intro to implementation details**)???

Services are logically separated into 5 different sub-modules: Challenges, Users, Questionnaires, Authentication and Execute Code.

**Challenges**)

The scope of the Challenges submodule includes the domains of Challenges, Challenge Answers, Challenge Tags and Tags.

Implementation wise each of these domains exposes a different Spring RestController and has a different service to handle the business logic. The possible operations exposed by these controllers can be found in the Swagger documentations (ref - 22).

The Challenge domain entity is represented by a combination of the data model entities Challenge and Challenge Solution, i.e. a challenge on the service module contains information not only about the challenge itself but also about its solutions.

The Challenge Answers domain entity is represented by a combination of the data model entities Challenge Answer and Answer, this is only natural since the Answer data model entity exists as part of a mandatory mutually exclusive relationship with different types of answers, i.e. every type of answer is related to an Answer data model entity.

The Challenge Tags domain entity is represented by a combination of the data model entity Tag and a record from the many to many entity CT, which represents the associations between a Tag and a challenge data model entities.

The Tags domain entity is represented by a data model entity Tag.

**Execute Code**)

The scope of the Execute submodule is very specific, its purpose is to receive requests to execute code remotely and if the request language is supported redirect the request to the execution environment module where the execution of the code will take place. The contract exposed by the controller of this submodule can be checked on the Swagger documentation (ref-22).

This submodule needs a property file to work properly named “executionEnvironments.properties”. This properties file has information about the endpoint to which the redirection of the execution requests is sent.