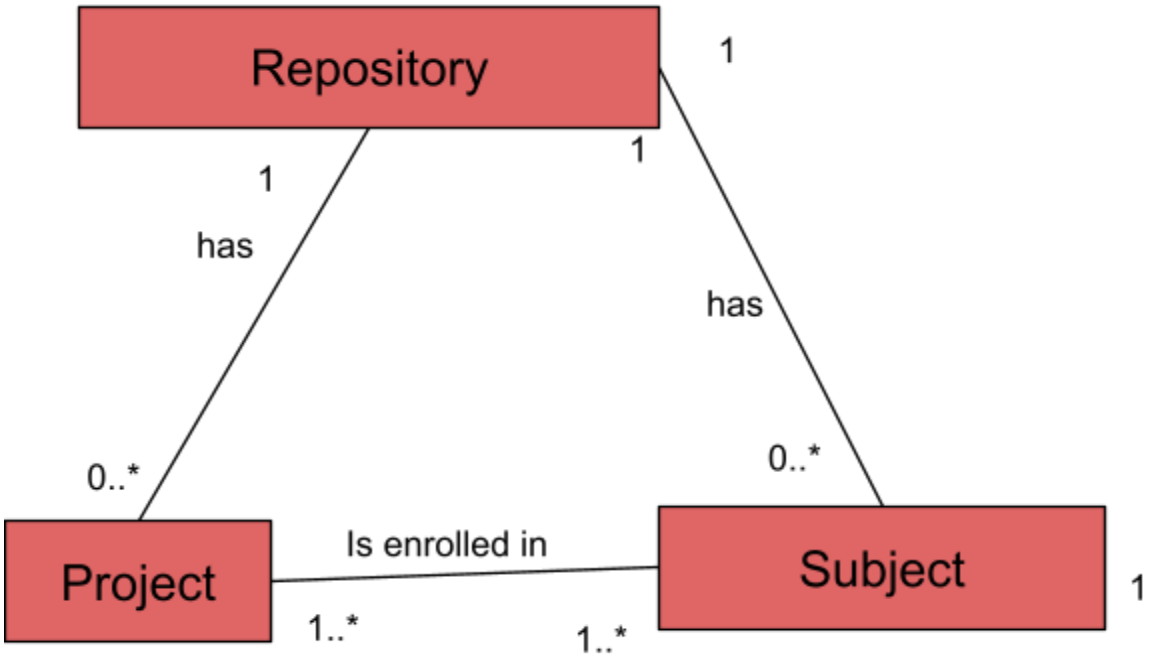
COLLECTIVE MINDS RADIOLOGY

PHP Back-End technical challenge

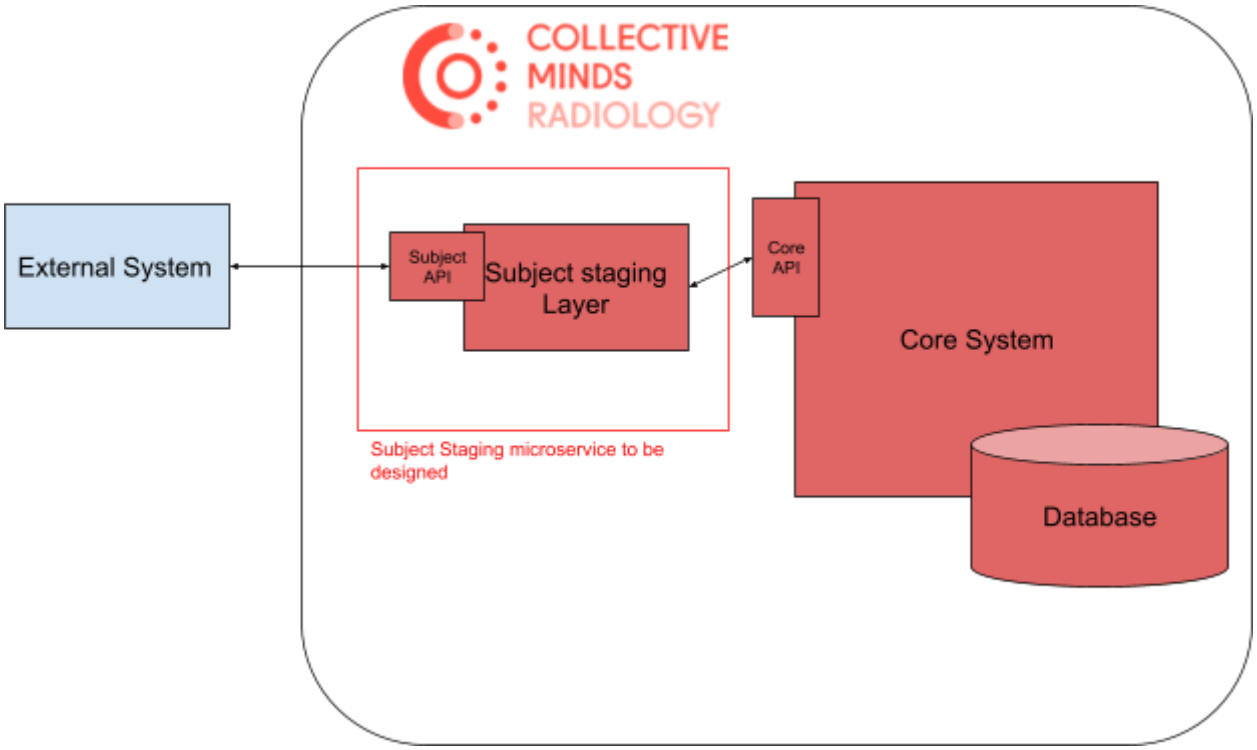
At Collective Minds we develop a Core platform that helps our customers manage data used in Clinical Trials. Those Clinical Trials are organized internally as “Projects” inside “Customer Repositories”, and each one of those repositories has a pool of individuals participating on different clinical trials. Those individuals (called “Subjects”) participate in a clinical trial either as a recipient of the investigational product(s) or as a control.



Our platform has to be able to connect/integrate to external systems used for Clinical Trials, that’s why we need to offer an API with endpoints for several relevant operations in CTs. Currently we have an existing Core API, which is supporting our Web Interface, but we don’t want to expose it directly to external systems. For that, we want to build a staging microservice (see figure below).

Focus of the first phase of the development of the staging layer is basic management of Subjects. For that reason we want to design a staging layer that exposes a **Subject API** (which is a subset of the **Core API,** see spec below) and fulfills the requirements that we describe in

the Requirements section in this document. In future releases we will be adding some logic to filter input data before reaching our Core platform. This is not to be covered here, but the architectural design shall be scalable and allow that.



The Core API looks currently like that:

Path: api.cmrad.com/API/v1

GET /repositories/{repositoryID}/subjects
GET /repositories/{repositoryID}/projects
GET /repositories/{repositoryID}/projects/{projectID}/subjects
GET /repositories/{repositoryID}/subjects/{subjectID}/projects
POST /repositories/{repositoryID}/projects/{projectID}
POST /repositories/{repositoryID}/subjects/{subjectID}
POST /repositories/{repositoryID}/projects/{projectID}/subjects/{subjectID}
POST /repositories/{repositoryID}/subjects/{subjectID}/projects/{projectID}

Requirements for the Subject staging layer and API:

- The external system shall be able to create subjects in the Collective Minds system through the staging layer
- The external system shall be able to assign projects to subjects in the Collective Minds system through the staging layer
- Duplicity of subjects inside the same repository is not allowed. Duplicated enrollment of subjects in projects shall not be allowed either.

Design guidelines

- PHP must be used as the main technology, feel free to choose a vanilla implementation or existing frameworks
- The staging layer should be implemented as a standalone microservice
- Filtering logic for subjects will be implemented in future releases. Architecture of the microservice shall support extendibility.
- No front-end/web access is required

Extra points

- TDD development is highly valued
- A containerized solution will make us very happy
- We forgot to add a detailed Core API specification based on a standard. It would be great if you can provide it for the Staging/Subject API that you design

Final words

- Requirements are deliberately vague. In case of doubt please make some educated guesses and justified assumptions. We are aware of the limitations of remote technical challenges and we are not looking for the perfect solution. We use this challenge to assess your reasoning, design skills and SW development processes.
- We know life is busy and finding time for technical challenges is often difficult. Hence, there is no deadline from our side but obviously the sooner you deliver the solution the better.
- We are **really** grateful for your interest in Collective Minds and the effort you spend on this challenge.