The project’s name is Call Center and it falls under NextCare namespace/package.

Project implemented in two completely different technologies (.Net & Java) however more emphasis is placed on the latter in terms of details, coverage standards, completeness etc..since I’ve already invested lot of time in .net ecosystem.

Same DB schema such as tables, relations, constraints, data types, indexes (almost), entities, business logic, functionality and endpoints. Ideally it is quite possible with few changes to have the client call the endpoints in both implementations with identical request i.e. same URI/payload.

Each project has its own conventions and standards based on its eco-system.

Database-wise: schema is simple yet it can be easily extended, indexes created where deemed necessary to speed up search; proper selection of data types available on the corresponding DB server along with its naming strategy

Following best design practices and security considerations our Entities are not exposed to clients. For this purpose and additional layer called ‘dto’ is added which as the name suggests contains the objects that are to be exposed to the outside world. This is extremely important to achieve loose coupling between the inner system and its clients particularly should the endpoints be externally available at one time.

Overall 5 APIs are provided (1 post & 4 get):

* Create a new claim (POST: api/claims) passing the payload in the request body
* Search claims (GET: api/claims?admitted-from=2020-10-31&admitted-till=2040-10-31&card-number=abc&hospital-id=1). all filters are optional so this works too (GET: api/claims) in this case however adding pagination functionality should be considered since it will return all claims.
* Find patient by their card no eg. GET: api/patients/abcdef0123456789. Used when the end user enters card no. and hits the apply button which would retrieve all the necessary information about the patient from the DB.
* Get all hospitals (GET: api/hospitals). Returns all hospitals. Usage is to populate the respective dropdown control in UI
* Get all physicians (GET: api/physicians). Returns all physicians. Usage is to populate the respective dropdown control in UI.

- Architecture wise, project is divided into 5 separate layers which can be applied to an enterprise-level business application (model -> repository -> dto -> service -> controller). Each layer providing a service to the one above it

Technologies/tools:

Java: IntelliJ’22, Java 17, spring boot, hibernate, JPA, JPQL, MySQL, spring web, micro service, rest api, integration testing, Json, Postman

.Net: VS’22, .net core 3.1, C# 8.0, EF core 5, LINQ, code first migration, micro services, rest API, SQL server, Json, Postman

-1 integration test for the creation of a new claim is provided (Java only)-

Explicitly specified how objects are to be serialized/de-serialized to conform to common conventions.