These are the domains to accommodate:

1. Candidate registration
2. Candidate profile

Candidate

- id: int

- email: String

- password: String

- profile: Profile

- active: boolean

Profile

- id: int

- photoPath: String

- givenName: String

- familyName: String

- dob: date

- height: int

- weight: float

- address1: String

- address2: String

- city: String

- province: String

1

1

We define two domains, Candidate and Profile, to accommodate the future development. And we will not store any image file in the database. The "photoPath: String" will store the path to the image file.

In most systems that have a User domain (or Candidate in this design), there are different roles for the User. For example, a User with an Administrator role will have access to features that are not available for a User who has the EndUser role. Or if we are designing a Role-playing game, there will be different roles with different stats for the player. To accommodate this requirement, the design should be:

Candidate

- id: int

- email: String

- password: String

- profile: Profile

- active: boolean

Profile

- id: int

- photoPath: String

- givenName: String

- familyName: String

- dob: date

- height: int

- weight: float

- address1: String

- address2: String

- city: String

- province: String

- role: Role

1

1

0..\*

1

Role

- id: int

- name: String

- modules: List<Module>

- stats: List<Stat>

- active: boolean

Then the system will also have the Module and Stat domains. The design with the Role domain will not be used for this back end application. For this application, we will use String for role and we can move all attributes from Profile to Candidate because of the one-to-one relationship.

Candidate

- id: int

- email: String

- password: String

- photoPath: String

- givenName: String

- familyName: String

- dob: date

- height: int

- weight: float

- address1: String

- address2: String

- city: String

- province: String

- role: String

- active: boolean