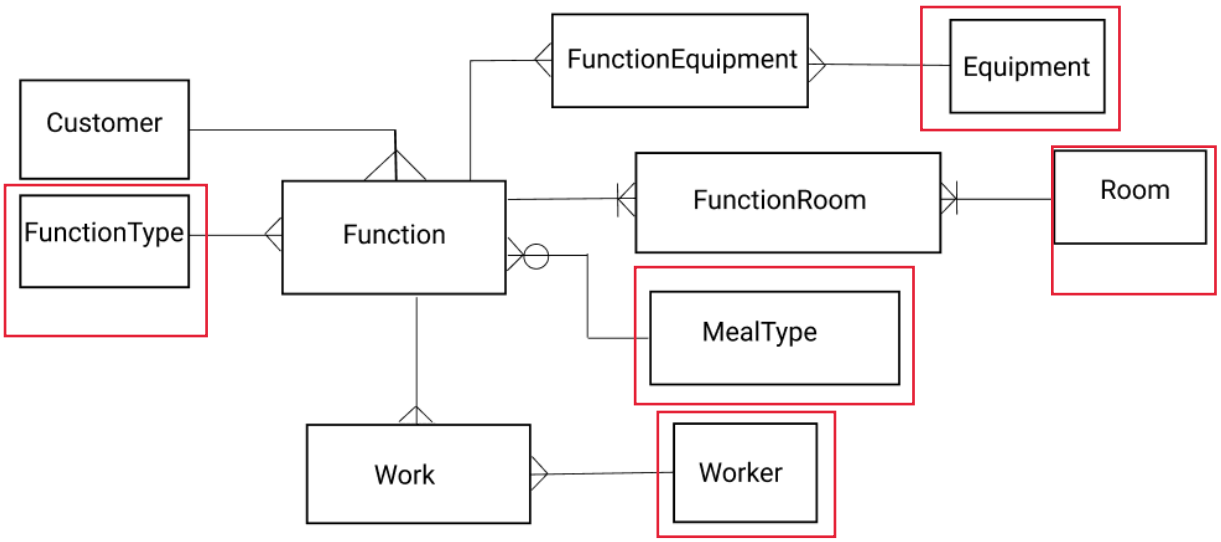
Instructions

Continue with the ASP.NET Core MVC Catering Management application from Part 2.
Please use the posted solution for Part 2 as the starting point for Part 3.



Please note that the additional classes for Worker, Work, Equipment and FunctionEquipment have already been added to the posted solution for Project Part 2, along with some seed data for each one.

Take time to examine the new classes and make sure you are familiar with the properties of each one.

13

1. (**2 Marks**) Make sure your name and student number are in comments at the top of your DbContext (CateringContext) class as well as having your name displayed on the Home/Index Page.
2. (**3 Marks**) DbSets for the new entities have already been added BUT you need to add one more thing in the Fluent API:
    a. Create a unique index on the combination of WorkerID and FunctionID in the Work entity. This is necessary to guarantee that a worker is not recorded as working more than once on a function.
    b. There are currently no migrations for the CateringContext in the posted solution. Start a new set of migrations and change your initializer to call `context.Database.Migrate();`
3. (**20 Marks**) You now have 5 "Lookup" entities to manage: Equipment, Room, MealType, Worker and FunctionType.
    a. (**10 Marks**) Create a Lookup Controller and Index view that will allow you to manage all of the lookup values from the one Index.
        i. You may use either of the two approaches taught in class.

b. (**10 Marks**) You will also need to create separate controllers and views for each of the lookup entities so they can be called from the Lookup Index to make the actual data modification.
   i. Make sure that you always navigate back to the Lookup Index after saving any changes to the data for a lookup value.
   ii. **BONUS:** make sure all code saving changes to the database is wrapped in try/catch blocks and that Edit Post actions use the TryUpdateModel approach.

14

4. **(20 Marks)** We need to be able to upload documents related to the Function, including Proposals, Contracts, Invoices etc.
   a. **(10 Marks)** A Function might need many documents, but each document relates back to a single Function.
      i. If a Function is deleted, all related documents should be automatically deleted as well (Cascade Delete).
      ii. While creating or editing a Function, the user should be able to select multiple files at once and upload them to the Function.
      iii. The user should be able to click links to download the related documents.
      iv. The Function/Details view should also show a list of all uploaded documents that relate to the Function with links to download them.
   b. **(10 Marks)** Add a new controller for managing the uploaded documents with the following:
      i. Index View showing the file names of the Documents along with the name of the Function it relates to.
      ii. Add filters for:
         1. Function with an All option at the top of the select element.
         2. Characters in the File Name.
      iii. Add paging but always sort by file name.
      iv. The Index should also have links for both downloading and deleting the documents.
      v. BONUS: Add a Description property to the Function Document and allow the user to edit both the File Name and Description from the new controller. Of course, descriptions cannot be added until after files have been uploaded.

You will **not** hand in Part 3A. This work is in preparation for the next steps in Part 3 on the MVC Catering Management application.