Minimum Viable Product Strategy

It is common to encounter situations in a project where you may not know what to do next. If that occurs, the guide below may help. It contains an outline of instructions which leverage code templates and github examples to execute the Create, Read, Update, and Delete operations for your MVP.

With practice, you will be able to begin and finish the code for the entire project in your head during the planning stage.

Models/Database Set Up

1. Create an MVC project with WebAPI functionalities (Updated -- Credits to Sandra)
    a. ASP .Net Web Application with C# (Not ASP.NET Core)
    b. Enter project name
    c. Select MVP
    d. Check the Web Api checkbox
    e. If you are going to have the user login then change the Authentication to be Individual User Accounts.
    f. There are two other files where you have to change your DB context name: App_Start/IdentityConfig.cs line 45 and App_Start/Startup.Auth.cs line 18

2. Install Entity Framework
    a. Tools > NuGet Package Manager > Manage NuGet Packages for solution > Browse > Search “entityframework” > 114M downloads, version 6.4.4
3. Create Models in your project with base information listed as fields.
    a. Use example code ‘Models/Player.cs’, ‘Models/Sponsor.cs’, /Models/Team.cs’ as a reference.
4. Use data annotations to specify the primary key field of each entity.
    a. Use example code ‘Models/Player.cs’, ‘Models/Sponsor.cs’, /Models/Team.cs’ as a reference.
    b. Make sure to also include the same references with at the beginning of the model (eg. System.ComponentModel.DataAnnotations)
5. Link the models together using code-first relationships.
    a. Use example code ‘Models/Player.cs’, ‘Models/Sponsor.cs’, /Models/Team.cs’ as a reference
6. Create a database context class in your Models folder
    a. Use example code ‘Models/VarsityDataContext.cs’ as a reference
7. Create a connection string property in the web.config file of your project
    a. Use example code ‘Web.Config’ as a reference
8. Enable Migrations for your project
    a. Tools > NuGet Package Manager > Package Manager Console
    b. Enable-migrations
9. Add a Migration
    a. Tools > NuGet Package Manager > Package Manager Console
    b. add-migration migration_name
    c. Each migration name must be unique.
10. Review the Migration
    a. Go to /Migrations/xxxxx_migration_name.cs
    b. Check the ‘Up’ methods
11. Update the database
    a. Tools > NuGet Package Manager > Package Manager Console
    b. Update-database
    c. If an error message occurs, read the error message and check the models again.

12. Check the database
    a. View > SQL Server Object Explorer
    b. Look under (localdb)/MSSQLLocalDB > Databases > Database Name
    c. Database name is defined by connection string property in web.config
    d. Seed data by right clicking table > View Data > Enter data manually
13. To apply future changes to the database, adjust your models and run steps 9-12 again

Controllers Set Up

1. Configure a WebAPI controller to perform CRUD operations
2. Begin by looking at some template code
    a. Controllers > New Controller > 'WebAPI Controller with read/write actions using entity framework'
    b. Select the model which you wish to perform CRUD operations on
    c. Select the database context you created
3. Customize the code to fit your own project needs
    a. Adjust the App_Start/webapiconfig.cs file to include the action attribute
    b. Customize the controllers to serve GET and POST requests
    c. Reference in-class example code /Controllers/PlayersController.cs
4. (optional) Utilize a data transfer object class (DTO) to exchange information instead of directly returning your Models.
5. Test that your WebAPI works as intended by sending CURL requests through the command prompt
    a. See the .readme for an example
6. Configure a Controller to access your WebAPI and serve Views.
7. Begin by looking at some template code
    a. Controllers > New Controller > 'MVC controller with read/write actions'
8. Your controller can access the WebAPI by directly invoking it, or sending a request.
    a. 5101 example of directly invoking the Web API controller
    b. 5204 example of sending a request to the Web API controller
9. Create methods which support server-rendered pages for Create, Read, Update, and Delete.
    a. List method for listing information about your entity
    b. Show method for showing one instance of your entity
    c. New method for showing a page that allows a user to enter new information about an entity
    d. Update method for showing a page that allows a user to adjust similar information about an entity
    e. Use your wireframes as a base for the pages you will need to create
    f. You can use the 5204 example as a reference for your MVP
    g. You can use the 5101 example as a reference for your MVP

Views Set Up

1. Create Views which match the controller methods you have created in Controllers steps 6-9
    a. Start by looking at some template code ‘Views’ > ‘New View’ > Choose Template
    b. Investigate the different templates and see how they would fit in the context of Create, Read, Update, Delete
2. Test the Views by reading data, creating data, deleting data, and updating data.
    a. Run the project and access each view through the browser.
3. Analyze your wireframes to determine the extra data needed to support your Views beyond the base information
4. Revisit your controllers to gain access to the data you need
    a. See examples of controllers which pull information from multiple sources in one method
    b. See example of a View Model to define information from multiple sources
    c. See example of a View which displays information from a View Model
    d. More examples coming soon
5. Analyze your wireframes to determine ways you can manipulate data in your system
6. Revisit your controllers to allow extra methods which add and remove relationships between your entities
    a. More examples coming soon