# SlackBot Group Project: ASP.NET Core Web API Backend Design

Team members: Tony Seichter, Alexander Fowle, Randall Gray (2020)

## Introduction

This document provides a high-level design of the ASP.NET web API created for the devCodeCamp SlackBot group project. The REST API was implemented using the Data Repository design pattern. This design pattern isolates the database implementation from the API controllers allowing easy swap out.

The database is in the Repository project. The ASP.NET web API is in the SearchHelperBot project. The API was implemented with a single controller, ValuesController, for simplification. All calls from the controller to the database are asynchronous. In addition to the API calls, the controller also contains calls to a language parser to improve user queries.

## Repository

### Models

(See <EntityRelationshipDiagram.drawio>)

The following models are used by the language parser to improve the user queries.

Several models contain a Day field. This field specifies the class day number after which the associated topic should be included with the search. This prevents searching on advance topics until they are covered in class. Currently all “Day” fields are set to 0.

(Class ids are “class name + Id”.)

**ActiveProject** {string ProjectType, int Day} (e.g. “asp.net mvc”)

**Language** {string LanguageName} (e.g. “c#”, “javascript”)

**Platform** {string PlatformName} (e.g. “.net”, “.net core”}

**PreferredLanguage** {string LanguageName} (e.g. “c#”)

**PreferredSearch** {string SearchName} (e.g “docs.microsoft.com”, “w3schools”)

These models have their main field appended to the end of the search strings to improve

searches.

**BadWord** {string Word} (e.g. “I”, “my”)

**BadPhrase** {string Phrase} (e.g. “trying to”, “want to”)

These models represent words and phrases that are removed from the search strings to improve searches.

**NearConceptPhrase** {string Phrase, [FK] int ConceptId} {e.g. “get values in”}

**NearConceptIdea** {string ProperForm} {e.g. “loop over each value”}

These models together define a Dictionary <Key=Phrase, Value=Idea>. There may be several Phrases linked to a single Idea. When a Phrase is found in a search string, it is replaced with the Idea string to improve searches.

The following models are used by the API.

**RawSearches** {string StudentName, DateTime Timestamp, string Search}

This model is used to store all student searches. The table may be queried by student name, timestamp, or any of the appended values from the other models to see where additional class help may be needed.

**Settings** {string SettingName, bool Set} (e.g. “Logging”, “false”}

This model holds fields that can be turned on or off to affect the functionality of the language parser. (The addition of settings requires functional implementation.)

### Other Classes

Every model has a repository class and a repository interface class. Every model repository class inherits from a repository base class and the model repository interface class. Every model repository interface class inherits from a repository base interface class. The repository base class and its interface are generic classes that can handle any of the models. The repository base class (and its interface class) implement the CRUD operations on the repository tables. Model specific methods may be added to the model repository (and interface) classes.

A RepositoryWrapper class ties all of the model repositories together and functions as an abstraction of the ApplicationDbContext class.

## SearchHelperBot

The SearchHelperBot consists of a single controller, two models for the controller input and output data, the language parser, and the language parser helper.

### ValuesController

In order to simplify the API, all calls to the controller are to the Post() method (single endpoint). The structure passed to the Post() method determines whether a language parse or a CRUD function - PostAdd(), PostGet(), PostPut(), PostDelete() - is to be performed. The data repository acted upon is also specified by the Post() input structure. The Post() method always returns a JSON string – a list of suggested search queries; or, the PostGet() result after data repository manipulation.

### Incoming Class (JSON)

{"search": {

"role": "student/instructor", // “student” => query search; “instructor” => CRUD

"username":"name", // RawSearch.StudentName

"request": {

"type": “=>”, // “add”-PostAdd(), “edit”-PostPut(), “remove”-

// PostDelete(), “<model name>”-PostGet()

"day": 0, // current class day to regulate query (future)

"search": "<user query>"

},

"add": { // PostAdd() input

"type": "<model name>",

"id": 0,

"name": "<data being added>",

"matchTo": "<NearConceptIdea text being matched to>”, // add.type IN

// “nearconcepts”,

// “nearconceptphrases”

"conceptKey": "0",

"day": 0 // class day this becomes relevant (future)

},

"edit": { // PostPut() input

"type": "<model name>",

"id": 0, // id of record being updated

"originalname": "",

"newname": "<updated data>", // updated data value

"matchto": "<NearConceptIdea text being matched to>", // See notes below.

"conceptKey": "",

"day": 0 // class day this becomes relevant (future)

},

"remove": { // PostDelete() input

"type": "<model name>",

"id": 0, // id of record being deleted

"name": ""

},

"setting":{

"id": 0, // id of setting for edit and delete, PostAdd=>add.name

"set": true // value to set setting

}

}}

**Note:** NearConceptPhrases/NearConceptIdeas

PostGet():

NearConceptIdeas can be retrieved by themselves.

NearConceptPhrases are retrieved with their NearConceptIdeas.

PostAdd():

NearConceptIdeas can be added by themselves.

NearConceptPhrases must be added with a NearConceptIdea. If the NearConceptIdea is not

already in the database, it is added and linked; otherwise, the existing NearConceptIdea is

linked.

PostPut():

NearConceptIdeas can be edited by themselves.

NearConceptPhrases must be edited with a NearConceptIdea. The NearConceptIdea is

not edited. If the NearConceptIdea is not already in the database, it is added and linked; otherwise, the existing NearConceptIdea is linked.

PostDelete():

NearConceptIdeas cannot be deleted.

NearConceptPhrases can be deleted. The linked NearConceptIdea remains in the database even

if it is no longer linked to anything.

### Outgoing Class

The outgoing class has member variables to return Lists of any of the models as specified by the responseType member variable.

### Language Parser

The language parser, SearchHelper.cs, takes in the user’s search query and returns a set of queries to improve the user’s search results. Words are removed, replaced, and appended to the search query based on the entries in the language parser database tables. The search helper, DbHandler.cs, converts the entries from the language parser database tables into more useable forms for the parser.