**Unit 14 Sequelize Homework: Reverse Engineering Code**

**Description and Purpose of Files**

**Develop Directory**

* server.js

Description/Purpose: This is the file where we set all of our node package managers. Also setup are the ‘requires’ for routes and information to use the express package for middleware, as well as handlebars. This is also the file that we use to start our server when ready to run the application.

* package.json

Description/Purpose: This file holds information for our node package managers. In this file, the node handlebars boilerplate is the name, and in order to begin the script, we need to use node server.js This file also includes version and dependency information for our node package manager. We can also see es-lint is used for linting as well as Sequelize as part of the database.

* .travis.yml

Description/Purpose: This is the file that is telling our Travis CI what to do with the project. It includes information for the language, which is node\_js as well as branches and directories to be used with Travis CI, and when to use them.

* .gitignore

Description/Purpose: This file tells which files in our project directory should not be pushed into the github repository. It usually includes sensitive information such as password information, or node modules which are large packages that do not need to be pushed to our github. This file also includes the npm debug and env. Files which also should not need to be pushed to the github repository.

* .eslintrc.json

Description/Purpose: This file is designed to define the configuration structure and allow us to configure information such as he environment and rules. In this file, all environment variables listed have been set to true, and rules have been setup.

* .eslintignore

Description/Purpose: This file tells our linter to ignore the files included in this file. Here we have told ESLint to ignore models/index.js and node\_modules when linted.

**Views Directory**

* index.handlebars

Description/Purpose: This is the handlebars file used to populate the html structure of our homepage when rendered. It includes all of the information needed to generate the simple home page to display text and provide forms to be filled out by the user.

* example.handlebars

Description/Purpose: This is the handlebars file used to setup the html structure of our page once a user enters text into he Example Text form, and Example Description, and then clicks ‘Submit’. This page displays the information that is created into the database for this text, such as the ID, text and Description.

* 404.handlebars

Description/Purpose: This is the handlebars file used to setup the html structure of our page when there is an error with the application or server.

**Views/Layouts Directory**

* main.handlebars

Description/Purpose: This is our simple handlebars file which is simply used to create the structure of our html. The handlebars files are used to generate information which will go into the body section of this file.

**Test Directory**

* canary.test.js

Description/Purpose: This JavaScript file, as noted, as setup as a canary test, which is one we setup to always pass. This helps to ensure our overall testing suite is setup properly and working before we begin writing real tests. In this file, chai is required and it is expected to pass this canary test.

**Routes Directory**

* htmlRoutes.js

Description/Purpose: This JavaScript file establishes the necessary routes for html. In it, we require the models and initially load the index page. We then load the example page and pass in an example by ID. Finally, if there is an error, we render a 404 page for any unmatched routes.

* apiRoutes.js

Description/Purpose: This JavaScript file establishes the necessary routes for getting, creating and deleting the examples on the page. We require the models and then establish the required JavaScript in order to get all examples, then create a new example, and then delete an example by ID.

**Public Directory**

**Public/Styles Directory**

* style.css

Description/Purpose: This is the file used to style our webpage. This file will link styles with specific classes or IDs established in our html. In this example, only a very simple style was rendered for the list-group-item class to adjust the line height to 2.5.

**Public/js Directory**

* index.js

Description/Purpose: This is the JavaScript file used to help render our public showing page, and allow users to interact with the page. First, we establish the references to page elements. We then setup the API object which contains methods for each kind of request made. For instance, the POST method was used and took example data and then JSON.stringified that data. We then create a function to refresh examples, which essentially gets new examples from the database and repopulates the list with those examples. A function was created to handle the form submissions and save them to the database when submitted, as well as refresh the list. Then a function was created to delete the example when the delete button is clicked. This was done by simply passing that example into the function, and subsequently deleting it and the data, including ID associated with that example. Finally, event listeners were added to submit and delete buttons so that when a user “clicks” them, the site runs those created functions.

**Models Directory**

* schema.sql

Description/Purpose: This is the file used to create our database. In this example, we are dropping the exampledb database if it exists, and then creating the database called exampledb. We also drop testdb if it exists, and then create the database testdb, essentially as a test database for the project.

* index.js

Description/Purpose: This JavaScript file is created to require all of our stored models. In this example, we have required fs, path and Sequelize, as well as called our configuration files in order to establish the requirements for connection to the database from the files.

* example.js

Description/Purpose: In this JavaScript file, we establish the export of data from an example to the database, and then created the function to return the example.

**Config Directory**

* config.json

Description/Purpose: This JSON file defines our application server side settings. We have established environments for development, test and production. In each of those environments, there is a username, which is usually “root”. The password is then entered in, which is unique to the host, which is setup as localhost. Finally, the dialect is mysql as that is the database that we are using.