OWL FRONT END/BACK END PROTOCOL DOCUMENT

This document describes the current the JSON object creation as the back end currently utilizes them.

Query

Queries will use the “POST” command in REST.

# Input

Inputs shall take the form of the database fields in the PDF document. Fields which are not filled may be omitted.

**{“PlantId”:value,”Name”:”value”,”ColorFlower”:”value”,”ColorFoliage”:”value”,”ColorFruitSeed”:”value”,”TextureFoliage”:”value”,”Shape”:”value”,””Pattern”:”value”,”ImageURL”:”value”,”USState”:”value”,”Type”:”value”,”Description”:null}**

**Or**

**{ColorFlower”:”value” }**

Where any field can be used, not just ColorFlower.

# Output

## No entry:

The back end shall return closed brackets [] (empty list)

Example:

**[]**

## Single Entry:

The back end shall return a JSON object inside a list (an object enclosed in brackets)

Example:

**[{“PlantId”:value,”Name”:”value”,”ColorFlower”:”value”,”ColorFoliage”:”value”,”ColorFruitSeed”:”value”,”TextureFoliage”:”value”,”Shape”:”value”,””Pattern”:”value”,”ImageURL”:”value”,”USState”:”value”,”Type”:”value”,”Description”:null}]**

(Note description is currently always null).

## Multiple Entries

The back end shall return JSON objects delimited by a comma inside the list.

**[{“PlantId”:value,”Name”:”value”,”ColorFlower”:”value”,”ColorFoliage”:”value”,”ColorFruitSeed”:”value”,”TextureFoliage”:”value”,”Shape”:”value”,””Pattern”:”value”,”ImageURL”:”value”,”USState”:”value”,”Type”:”value”,”Description”:null},{“PlantId”:value,”Name”:”value”,”ColorFlower”:”value”,”ColorFoliage”:”value”,”ColorFruitSeed”:”value”,”TextureFoliage”:”value”,”Shape”:”value”,””Pattern”:”value”,”ImageURL”:”value”,”USState”:”value”,”Type”:”value”,”Description”:null}]**

## Error Condition

The back end shall return the message “Invalid Message!” in the event that it does not recognize the message.

Field Return

Fields will be returned for each question in the database. A “GET” command in REST will return the questions.

# Input

The input will be a message in the header of a GET of “Id=Fields”.

# Output

The output will be a list of FieldObjs, which will be JSON objects of the following form:

**[{“Column”=”value”, “Fields”:[“value1”,”value2”,”value3”,...]}, {“Column”=”value”, “Fields”:[“value1”,”value2”,”value3”,...]},...]**

Each FieldObj stores a string identifying the column and the fields of the database (values as options). If there is any error, the system shall return **Error! Fields could not be returned**

Login

The login will be stored as two fields added to the FloraObj: UserName and Password. To perform a login, the user will send a “PUT” command with no fields except for UserName and Password defined. The system shall interpret this as a login request, and shall return **Success Login** if the login is valid or **Error-Login Invalid!** if the login failed.

The front end shall pass in the following for a login:

**{“UserName”=”value”, “Password”=”value” }**

Insertion/Updates

Insertion will work with the “PUT” command in REST.

# Input

Inputs shall take the form of the database fields in the PDF document. Fields which are not filled will have no value. If the field PlantID is included, the system shall attempt to edit an existing entry. If the entry does not exist, the plant shall be created. The fields UserName and Password will need to be included as part of the insertion as there is no state in

**{“PlantId”:value,”Name”:”value”,”ColorFlower”:”value”,”ColorFoliage”:”value”,”ColorFruitSeed”:”value”,”TextureFoliage”:”value”,”Shape”:”value”,””Pattern”:”value”,”ImageURL”:”value”,”USState”:”value”,”Type”:”value”,”Description”:null, “UserName”=”value”, “Password”=”value” }**

**Or**

**{“PlantId”:value, ColorFlower”:”value”, “UserName”=”value”, “Password”=”value” }**

Where any field can be used, not just ColorFlower.

# Output

The Output will be **Error-Insertion Failed!** if there is a failure, **Error-Login Invalid!** if the login failed, or the full JSON object (all fields that were populated returned) if success.