**Documentation – causes and recommendations component**

**General info:**

Note: details of implementation are commented in code itself

Routing:

This application uses the UI-Router routing framework, and the routing for everything lies in config.js.

Dashboard:

The dashboard lies in views/index.html (note: do not confuse with index.html in the root folder, which contains css, plugin links, js scripts etc.), and has 6 controllers:

controllers.js – main controller, controls the toggling and triggers toggle event to the 5 child controllers below

controller.db.carbon.js – controls the carbon footprint panel

controller.db.hotspots.js – controls the hotspots (aggregated and normalised) panels

controller.db.processEffi.js – controls the process efficiency charts panel

controller.db.wasteAmt.js – controls the absolute waste amount bar chart panel

controller.db.wasteCost.js – controls the cost of waste panel

Hierarchy mod: (hierarchy.js)

Encompasses the above 5 child controllers, as well as the ones below:

controller.hotspoteval.timed.js (base controller)

* base controller of checklist and flowchart, this controls the measuresRec.html page
* draws bar chart of processes and hotspots that is gotten from dataService service (i.e. passed over from the hotspot.js controller)
* draws pie chart of the wastes in the process after user clicks on a particular process bar in the processes bar chart and assigns clicked process to $scope.selectedProcess
* assigns clicked waste to $scope.selectedWaste when user clicks on a particular slice of the drawn pie chart of the current selected process
* **\*** when user finishes evaluation and saves their results (by clicking save in the flowchart popup), this page will save the results by getting results stored in the sharedProperties service
* when user clicks ‘evaluate’ button, if waste has been evaluated before, will grab previous results from database (this will already be grabbed when init() is called at the start) and store in sharedProperties service for checklist.js to retrieve and display when the checklist pop up is opened.

controller.checklist.js

* contains $scope.edibleCausesList object, containing full list of causes
* controls the checklist.html popup, will open when user clicks ‘evaluate’ button on the base controller’s page
* flowchart pop up is opened up from within this controller when user clicks ‘find recommendations’ button
* retrieves previous results from sharedProperties service if waste has been evaluated before
* read documentation on checklist in **Documentation for causal analysis checklist.docx**

controller.flowchart.js

* controls the flowchart.html popup, will open when user clicks ‘find recommendations’ button in the checklist popup
* does not retrieve previous recommendations results from sharedProperties service as user should be required to go through flowchart again (doesn’t make sense to start somewhere halfway since they are re-evaluating and hence should be going down a new path from the start in order to be able to access all possible routes)
* saves evaluation results when user clicks ‘save results’ by storing it in sharedProperties service, and upon returning to controller.hotspoteval.timed.js, refer to **\*** under controller.hotspoteval.timed.js for explanation
* read documentation on flowchart in **Documentation for hierarchal flowchart.docx**
* read documentation on animal feed checklist in **AFC.vsdx** (open using Visio)
* look at flowchart layout in **FLW flowchart.vsdx** (open using Visio)

sharedProperties service:

* in services.js
* acts as a temporary storage to transport information between base controller, checklist and flowchart controllers

Viewing causes and recommendations after evaluation:

* pop-ups: causes.html and recommendations.html
* no assigned controller, only passes in scope from base controller using ngDialog.open()
* only works when wasteViewing is not null, so wasteViewing is assigned from selectedWaste when either ‘causes’ button or ‘recommendations’ button is clicked

Constants:

* in constants.js
* contains enum FLWTYPES, which contains definitions of the 3 types of food wastes defined in accordance to the taxonomy
* used as tooltip for definition of food waste (upon determination of food waste type) in checklist pop up

Others:

*scoringsystem.js:*

* obsolete, might be recovered in future if scoring system in the food waste standard gets restored
* mapped to scoringsystem.html

*hotspotseval.timeless.js*

* obsolete, might be recovered in future if team decides to implement feature where user can choose to add random evaluation results without identifying hotspots through our hotspots component