```r
########################################################################################################################################################
#############-------------------------------------------------------------------------------------------------------------------------------
#############
#############-------------------------------------------- Setup Module 2: Construct hierarchy ---------------------------------------------------
#############
#############-------------------------------------------------------------------------------------------------------------------------------
#############
########################################################################################################################################################

#'
#' SUMMARY:
#'
#' MODULE MAP:
#'
#' UI
#'
#' SERVER
#'
#' KEY COMPONENTS:
#'
#' NOTES:
#'


#####################################################################################################################################################


weightmodInput <- function(id, items) {

  ns <- NS(id)

  IDs <- items %>% length %>% seq_len

  column(12,

         fluidRow(

           uiOutput(ns("pie"))

         ),
         fluidRow(

           column(2, h4("Weight"), br(), if(length(IDs) > 2) checkboxInput(ns("visual"), label = "Visual?")),

           column(10,

                  fluidRow(

                    lapply(IDs, function(x) {

                      column(3,
                             fluidRow(
                               h4(items[x])
                             ),
                             fluidRow(

                               actionButton(ns(paste0("goplus_", x)), label = h4("+  "), width = "35px"),

                               actionButton(ns(paste0("gominus_", x)), label = h4("-  "), width = "35px"),

                               textOutput(ns(paste0("weight_", x)))

                             )
                      )

                    })

                  )

           )

         )

  )


}
#####################################################################################################################################################

weightmod <- function(input, output, session, items, level) {

  ns <- session$ns

  IDs <- items %>% length  %>% seq_len

  observe({

    req(input$visual)

    "weightmod (items):" %>% print()
    items %>% print

    "weightmod (input$visual):" %>% print
    input$visual %>% print

    output$pie <- renderUI({

      #' Save data frame used to generate pie chart, depending on reactive value rv$weights.

      if ((length(IDs) > 2) & (input$visual == TRUE)) {

        plotOutput(ns("piechart"))

      } else {

      }

    })

  })

  "Experiment" %>% print
  location <- (ns("") %>% strsplit(., split = "weightmod-"))[[1]]
  location %>% print
  A$rv[[as.character(level)]]$Location %>% print
  rv <- reactiveValues()

  #' Location booleans

  NameIndex <- A$rv[[as.character(level)]]$Name %in% items
  LocationIndex <- A$rv[[as.character(level)]]$Location %in% location
  OverallIndex <- NameIndex & LocationIndex

  rv$weights <- A$rv[[as.character(level)]]$Weight[OverallIndex]

  weightIDs <- rv$weights %>% length %>% seq_len

  lapply(IDs, function (x) {

    #' Observer adds some to target category weight but controls sum of category weights to remain
    #' equal to 1.

    observeEvent(input[[paste0("goplus_", x)]], {

      #' Add .01 to weight of category with ID == x

      rv$weights[weightIDs == x] <- rv$weights[weightIDs == x] + .01

      #' Subtract what was added to previous category to overall, but from all areas equally

      for (i in weightIDs[!(weightIDs %in% x)]) {

        rv$weights[weightIDs == i] <- rv$weights[weightIDs == i] - (.01)/(length(weightIDs)-1)

      }

      A$rv[[as.character(level)]]$Weight[OverallIndex] <- rv$weights

    })

  })

  #' Use IDs to generate '-' observers corresponding to every category

  lapply(IDs, function (x) {

    #' Observer subtracts some from target category weight but controls sum of category weights to remain
    #' equal to 1.

    observeEvent(input[[paste0("gominus_", x)]], {

      #' Subtract .01 to weight of category with ID == x

      rv$weights[weightIDs == x] <- rv$weights[weightIDs == x] - .01

      #' Add what was subtracted from previous category to overall, but from all areas equally

      for (i in IDs[!(IDs %in% x)]) {

        rv$weights[weightIDs == i] <- rv$weights[weightIDs == i] + (.01)/(length(weightIDs)-1)

      }

      A$rv[[as.character(level)]]$Weight[OverallIndex] <- rv$weights

    })

  })

  lapply(weightIDs, function(x) {

    output[[paste0("weight_", x)]] <- renderText({

      rv$weights[x]

    })

  })

  output$piechart <- renderPlot({

    dfT <- data.frame(subjects = items, value = rv$weights)

    ggplot(dfT, aes(x = "", y = value, fill = subjects)) + geom_bar(width = 2, stat = "identity") +
      coord_polar("y", start=0) + scale_fill_manual(values = colfunc(length(IDs))) +
      theme(axis.title.x = element_blank()) + theme(axis.title.y = element_blank())

  })


}

##########################################################################################################################################################

recursiveModuleInput <- function(id, ID, level) {

  ns <- NS(id)

  nextlevel <- level + 1

  paste0("ID: ", ID) %>% print
  paste0("level: ", level) %>% print

  ns("submit") %>% print

  Title <- (A$rv[[as.character(level-1)]][["Names"]])[as.numeric(ID)]

  box(width = 12, title = h3(Title), status = "primary",
      column(10, offset = 1,

             fluidRow(textInput(ns( paste0("i", level, "_items") ), label = "Type more:")),
             fluidRow(actionButton(ns("submit"), label = "Submit")), br(),
             fluidRow(uiOutput(ns("ui")))

      )
  )

}

recursiveModule <- function(input, output, session, ID, level) {

  nextlevel <- level + 1
  ns <- session$ns

  ns("testing_server") %>% print

  observeEvent(input$submit, priority = 2, {

    rv <- reactiveValues()

    req(input[[paste0("i", level, "_items")]])

    items <- VectorizeString(input[[paste0("i", level, "_items")]])

    locationVector <- strsplit(ns(""), "-")[[1]] %>% as.numeric

    ########################################################################################

    if (is.null(A$rv[[as.character(level)]])) {
      A$rv <- list(list(Names = c(), Location = c(), Weight = c())) %>% append(A$rv, . )
      names(A$rv)[length(A$rv)] <- as.character(level)
    }

    "Focus your attention here: " %>% print()
    items %>% print
    A$rv[[as.character(level)]][["Names"]] %>% print

    rep(ns(""), length(items)) %>% print
    A$rv[[as.character(level)]][["Location"]] %>% print

    NamesIndex <- items %in% A$rv[[as.character(level)]][["Names"]]
    LocationIndex <- rep(ns(""), length(items)) %in% A$rv[[as.character(level)]][["Location"]]
    OverallIndex <- !NamesIndex | !LocationIndex

    initialWeights <- OverallIndex %>% length %>% rep( 1/. , . )

    A$rv[[as.character(level)]][["Names"]] <- append(A$rv[[as.character(level)]][["Names"]], items[OverallIndex])
    A$rv[[as.character(level)]][["Location"]] <- append(A$rv[[as.character(level)]][["Location"]], rep(ns(""), length(items))[OverallIndex])
    A$rv[[as.character(level)]][["Weight"]] <- append(A$rv[[as.character(level)]][["Weight"]], initialWeights)

    #########################################################################################

    #' quick hack

    numberOfNewItems <- items[OverallIndex] %>% length
    totalNumberOfItems <- A$rv[[as.character(level)]][["Names"]] %>% length
    difference <- totalNumberOfItems - numberOfNewItems
    moduleIDsToCall <- (difference+1):(difference + numberOfNewItems)

    output$ui <- renderUI({

      Rows <- tagList()
      Rows[[1]] <- fluidRow(weightmodInput(ns("weightmod"), items = items[OverallIndex]))
      Rows[moduleIDsToCall+1] <- lapply(as.character(moduleIDsToCall),

                                        function(id) {

                                          recursiveModuleInput(ns(id), ID = id, level = level + 1) %>% fluidRow

                                        })
      Rows

    })

    for (id in as.character(moduleIDsToCall) %>% append( . , "weightmod")) {

      if (id != "weightmod") {

        callModule(recursiveModule, id, ID = id, level = level + 1)

      } else {

        callModule(weightmod, id = "weightmod", items = items[OverallIndex], level = level)

      }

    }

})

}

setup_hierarchyInput <- function(id) {

  ns <- NS(id)

  indexVec <- A$rv[["0"]]$Names %>% length %>% seq_len %>% as.character

  Title <- A$rv[["0"]]$Names[indexVec == id]

  Rows <- tagList()

  Rows[[1]] <- fluidRow(column(8, offset = 2, h2(Title) ))
  Rows[2:5] <- lapply(2:5, function(x) { br() })
  Rows[[6]] <- fluidRow(

    column(6, offset = 2, h4("Type your items into this box"),
           textInput(ns("i1_items"), label = "", width = "98%"))

  )
  Rows[[7]] <- fluidRow(

    column(6, offset = 2, actionButton(ns("submit"), label = "Submit"))

  )
  Rows[[8]] <- br()
  Rows[[9]] <- fluidRow(

    column(10, offset = 1, uiOutput(ns("ui")))

  )
  Rows[10:16] <- lapply(10:16, function(x) { br() })

  Overall <- tagList()
  Overall[[1]] <- br()
  Overall[[2]] <- fluidRow(column(10, offset = 1, Rows))
  Overall
```

```r
}

setup_hierarchy <- function(input, output, session, proceed, ID) {

  ns <- session$ns

  observeEvent(input$submit, {

    req(input$i1_items)

    items <- VectorizeString(input$i1_items)

    #########################################################

    if (is.null(A$rv[["1"]])) {
      A$rv <- list(list(Names = c(), Location = c(), Weight = c())) %>% append(A$rv, . )
      names(A$rv)[length(A$rv)] <- "1"
    }

    NamesIndex <- items %in% A$rv[["1"]][["Names"]]
    LocationIndex <- rep(ns(""), length(items)) %in% A$rv[["1"]][["Location"]]
    OverallIndex <- !NamesIndex | !LocationIndex

    initialWeights <- OverallIndex %>% length %>% rep( 1/. , . )

    A$rv[["1"]][["Names"]] <- append(A$rv[["1"]][["Names"]], items[OverallIndex])
    A$rv[["1"]][["Location"]] <- append(A$rv[["1"]][["Location"]], rep(ns(""), length(items))[OverallIndex])
    A$rv[["1"]][["Weight"]] <- append(A$rv[["1"]][["Weight"]], initialWeights)

    #######################################################

    #' quick hack

    numberOfNewItems <- items[OverallIndex] %>% length
    totalNumberOfItems <- A$rv[["1"]][["Names"]] %>% length
    difference <- totalNumberOfItems - numberOfNewItems
    moduleIDsToCall <- (difference+1):(difference + numberOfNewItems)

    output$ui <- renderUI({

      Rows <- tagList()
      Rows[[1]] <- fluidRow(weightmodInput(ns("weightmod"), items = items[OverallIndex]))
      Rows[moduleIDsToCall+1] <- lapply(as.character(moduleIDsToCall),

                                        function(id) {

                                          recursiveModuleInput(id = ns(id), ID = id, level = 2) %>% fluidRow

                                        })
      Rows

    })

    for (id in as.character(moduleIDsToCall) %>% append( . , "weightmod")) {

      if (id != "weightmod") {

        callModule(recursiveModule, id, ID = id, level = 2)

      } else {

        callModule(weightmod, id = "weightmod", items = items[OverallIndex], level = 2)

      }

    }

  })

  observeEvent(proceed(), priority = 3, {

    A$rv %>% print

    iterator <- A$rv %>% length %>% seq_len

    if (length(iterator) <= 2) {

    } else {

      A$iH <- lapply(iterator, function(i) {

        items <- A$rv[[i]]$Names
        location <- A$rv[[i]]$Location
        weight <- A$rv[[i]]$Weight

        uniqueLocations <- unique(location)

        if (i == 1) {

          IDColumns <- matrix(nrow = items %>% length, ncol = 0) %>% as_tibble()

          tibble(
            ID = items %>% length %>% seq_len %>% as.list ,
            Name = items %>% as.list,
            Notes = items %>% length %>% rep("", . ) %>% as.list,
            Extent = items %>% length %>% rep("", . ) %>% as.list,
            Start = items %>% length %>% rep(Sys.time(), . ) %>% as.list,
            End = items %>% length %>% rep("", . ) %>% as.list
          ) %>% cbind( . , IDColumns) %>% as_tibble %>% as.data.frame

        } else {

          datframes <- lapply(uniqueLocations, function(j) {

            rows <- location %in% j %>% sum

            locationVector <- strsplit(j, "-")[[1]] %>% as.numeric

            IDColumns <- sapply(locationVector, function(x) { rep(x, rows) })

            IDColumns <- if (class(IDColumns) != "matrix") as.matrix(IDColumns) %>% t() %>% as_tibble else IDColumns %>% as_tibble

            IDColumns[] <- lapply(1:ncol(IDColumns), function(x) {

              IDColumns[[x]] <- as.list(IDColumns[[x]])
            })

            IDColumns

          })

          if ((class(datframes) == "list") & (length(datframes) == 1)) datframes <- datframes[[1]]

          if ((length(datframes) == 1) | all(class(datframes) != "list")) IDColumns <- datframes else IDColumns <- do.call(rbind, datframes) %>% as_tibble

          colnames(IDColumns) <- sapply(0:(i-2), function(x) {paste0("i", x)})

          tibble(
            ID = items %>% length %>% seq_len %>% as.list ,
            Name = items %>% as.list,
            Notes = items %>% length %>% rep("", . ) %>% as.list,
            Extent = items %>% length %>% rep("", . ) %>% as.list,
            Weight = weight %>% as.list,
            Start = items %>% length %>% rep(Sys.time(), . ) %>% as.list,
            End = items %>% length %>% rep("", . ) %>% as.list
          ) %>% cbind( . , IDColumns) %>% as_tibble %>% as.data.frame

        }


      })

    }


  })


}
```