app.R

ndeke

2022-06-05

library(ECharts2Shiny)  
library(waiter)  
library(shiny)  
library(plotly)

## Loading required package: ggplot2

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

## The following object is masked from 'package:stats':  
##   
## filter

## The following object is masked from 'package:graphics':  
##   
## layout

library(ggplot2)  
library(data.table)  
library(purrr)

##   
## Attaching package: 'purrr'

## The following object is masked from 'package:data.table':  
##   
## transpose

library(shinyjqui)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':  
##   
## between, first, last

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(echarts4r)  
library(readr)  
library(readxl)  
library(shiny)  
library(shinycssloaders)  
library(shinydashboard)

##   
## Attaching package: 'shinydashboard'

## The following object is masked from 'package:graphics':  
##   
## box

library(dashboardthemes)  
library(bslib)

##   
## Attaching package: 'bslib'

## The following object is masked from 'package:utils':  
##   
## page

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble 3.1.6 v stringr 1.4.0  
## v tidyr 1.2.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::between() masks data.table::between()  
## x dplyr::filter() masks plotly::filter(), stats::filter()  
## x dplyr::first() masks data.table::first()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::last() masks data.table::last()  
## x purrr::transpose() masks data.table::transpose()

library(shinydashboardPlus)

##   
## Attaching package: 'shinydashboardPlus'

## The following objects are masked from 'package:shinydashboard':  
##   
## box, dashboardHeader, dashboardPage, dashboardSidebar, messageItem,  
## notificationItem, taskItem

## The following object is masked from 'package:graphics':  
##   
## box

library(shinyWidgets)

##   
## Attaching package: 'shinyWidgets'

## The following object is masked from 'package:shinydashboardPlus':  
##   
## progressBar

library(shinyFeedback)  
library(shinyalert)

##   
## Attaching package: 'shinyalert'

## The following object is masked from 'package:shiny':  
##   
## runExample

library(shinyjs)

##   
## Attaching package: 'shinyjs'

## The following object is masked from 'package:shinyalert':  
##   
## runExample

## The following object is masked from 'package:shinyWidgets':  
##   
## alert

## The following object is masked from 'package:shiny':  
##   
## runExample

## The following objects are masked from 'package:methods':  
##   
## removeClass, show

library(png)  
library(later)  
library(DT)

##   
## Attaching package: 'DT'

## The following objects are masked from 'package:shiny':  
##   
## dataTableOutput, renderDataTable

library(rhandsontable)  
not\_sel <- "Not Selected"  
theme1 <- theme(  
 axis.line = element\_line(colour = 'grey50', size = .75),  
 panel.background = element\_rect(fill='lightblue'),  
 plot.background = element\_rect(colour='darkblue'),  
 panel.grid.major.y = element\_line(colour = "grey50"),  
 panel.grid.minor = element\_blank(),  
 panel.grid.major.x =element\_blank(),  
 axis.text = element\_text(colour = "darkblue"),  
 axis.title = element\_text(colour = "darkblue")  
)  
myToastOptions <- list(  
 positionClass = "toast-top-right",  
 progressBar = FALSE,  
 timeOut = 3000,  
 closeButton = TRUE,  
   
 # same as defaults  
 newestOnTop = TRUE,  
 preventDuplicates = FALSE,  
 showDuration = 300,  
 hideDuration = 1000,  
 extendedTimeOut = 1000,  
 showEasing = "linear",  
 hideEasing = "linear",  
 showMethod = "fadeIn",  
 hideMethod = "fadeOut"  
)  
  
exports<- openxlsx::read.xlsx('~/Programming/R/DATA/exports.xlsx',detectDates = TRUE)  
exports1<- openxlsx::read.xlsx('~/Programming/R/DATA/exports1.xlsx',detectDates = TRUE)  
kenya\_status1<- openxlsx::read.xlsx('~/Programming/R/DATA/kenya\_status1.xlsx',detectDates = TRUE)  
  
 source("exports\_modal\_dialog.R")  
source('debt\_modal\_dialog.R')  
dataset<-c('Exports','Kenya Debt','Remittances')  
disableActionButton <- function(id,session) {  
 session$sendCustomMessage(type="jsCode",  
 list(code= paste("$('#",id,"').prop('disabled',true)"  
 ,sep="")))  
}  
enableActionButton <- function(id,session) {  
 session$sendCustomMessage(type="jsCode",  
 list(code= paste("$('#",id,"').prop('disabled',false)"  
 ,sep="")))  
}  
  
customTheme <- shinyDashboardThemeDIY(  
 ### general  
 appFontFamily = "Candara"  
 ,appFontColor = "rgb(0, 0, 0)"  
 ,primaryFontColor = "rgb(0, 0, 0)"  
 ,infoFontColor = "rgb(8,58,68)"  
 ,successFontColor = "rgb(0, 0, 70)"  
 ,warningFontColor = "rgb(255,255,0)"  
 ,dangerFontColor = "rgb(255,0,0)"  
 ,bodyBackColor =cssGradientThreeColors(  
 direction = "down"  
 ,colorStart = "rgb(255, 240, 255)"  
 ,colorMiddle = "rgb(255, 240, 200)"  
 ,colorEnd = "rgb(255, 240,170)"  
 ,colorStartPos = 0  
 ,colorMiddlePos = 50  
 ,colorEndPos = 100  
 )  
   
 ### header  
 ,logoBackColor = "rgb(200, 60, 0)"  
   
 ,headerButtonBackColor = "rgb(255, 240, 255)"  
 ,headerButtonIconColor = "rgb(0,0,100)"  
 ,headerButtonBackColorHover ="rgb(50, 50, 50)"  
 ,headerButtonIconColorHover = "rgb(0,255,255)"  
   
 ,headerBackColor = "rgb(200, 60, 0)"  
 ,headerBoxShadowColor ="rgb(100, 100, 100)"  
 ,headerBoxShadowSize = "2px 2px 2px"  
   
 ### sidebar  
 ,sidebarBackColor = cssGradientThreeColors(  
 direction = "down"  
 ,colorStart = "rgb(0, 0, 50)"  
 ,colorMiddle = "rgb(0, 0, 80)"  
 ,colorEnd = "rgb(0, 0, 120)"  
 ,colorStartPos = 0  
 ,colorMiddlePos = 50  
 ,colorEndPos = 100   
 )   
 ,sidebarPadding = 0.5  
   
 ,sidebarMenuBackColor = ""  
 ,sidebarMenuPadding = 0.5  
 ,sidebarMenuBorderRadius = 0.5  
   
 ,sidebarShadowRadius = "2px 2px 2px"  
 ,sidebarShadowColor = "rgb(100,100,100)"  
   
 ,sidebarUserTextColor = "rgb(0,255,255)"  
   
 ,sidebarSearchBackColor = "rgb(189,54,121)"  
 ,sidebarSearchIconColor = "rgb(102,51,0)"  
 ,sidebarSearchBorderColor = "rgb(102,255,255)"  
   
 ,sidebarTabTextColor = "rgb(255, 255, 255)"  
 ,sidebarTabTextSize = 18  
 ,sidebarTabBorderStyle = "none none solid none"  
 ,sidebarTabBorderColor = "rgb(0, 255, 255)"  
 ,sidebarTabBorderWidth = 1  
   
 ,sidebarTabBackColorSelected =cssGradientThreeColors(  
 direction = "right"  
 ,colorStart = "rgb(255, 240, 255)"  
 ,colorMiddle = "rgb(255, 240, 200)"  
 ,colorEnd = "rgb(255, 240,170)"  
 ,colorStartPos = 0  
 ,colorMiddlePos = 50  
 ,colorEndPos = 100  
 )  
 ,sidebarTabTextColorSelected = "rgb(0, 0, 0)"  
 ,sidebarTabRadiusSelected = "0px 20px 20px 0px"  
   
 ,sidebarTabBackColorHover = cssGradientThreeColors(  
 direction = "right"  
 ,colorStart = "rgb(255,150, 0)"  
 ,colorMiddle = "rgb(225, 125, 0)"  
 ,colorEnd = "rgb(200, 100,0)"  
 ,colorStartPos = 0  
 ,colorMiddlePos = 50  
 ,colorEndPos = 100  
 )  
 ,sidebarTabTextColorHover = "rgb(0,255,255)"  
 ,sidebarTabBorderStyleHover = "none none solid none"  
 ,sidebarTabBorderColorHover = "rgb(200, 60, 0)"  
 ,sidebarTabBorderWidthHover = 1  
 ,sidebarTabRadiusHover = "0px 20px 20px 0px"  
   
 ### boxes  
 ,boxBackColor = "rgb(255,240,255)"  
 ,boxBorderRadius = 5  
 ,boxShadowSize = "0px 1px 1px"  
 ,boxShadowColor = "#ac001a"  
 ,boxTitleSize = 16  
 ,boxDefaultColor = "rgb(170,234,5)"  
 ,boxPrimaryColor = "rgb(138,268,183)"  
 ,boxInfoColor = "rgb(0,179,179)"  
 ,boxSuccessColor = "rgba(0,255,213,1)"  
 ,boxWarningColor = "rgb(0,179,179)"  
 ,boxDangerColor = "rgb(255,88,55)"  
   
 ,tabBoxTabColor = "rgb(44,255,255)"  
 ,tabBoxTabTextSize = 14  
 ,tabBoxTabTextColor = "rgb(2, 3, 102)"  
 ,tabBoxTabTextColorSelected = "rgb(90,252,154)"  
 ,tabBoxBackColor = "rgb(3,3,156)"  
 ,tabBoxHighlightColor = "rgba(44,222,235,1)"  
 ,tabBoxBorderRadius = 5  
   
 ### inputs  
 ,buttonBackColor = "rgb(255, 240, 255)"  
 ,buttonTextColor = "rgb(0,0,0)"  
 ,buttonBorderColor = "rgb(200, 60, 0)"  
 ,buttonBorderRadius = 5  
   
 ,buttonBackColorHover ="rgb(200, 60, 0)"  
 ,buttonTextColorHover = "rgb(0,255,255)"  
 ,buttonBorderColorHover = "rgb(118,118,102)"  
   
 ,textboxBackColor = ""  
 ,textboxBorderColor = "rgb(0,255,255)"  
 ,textboxBorderRadius = 5  
 ,textboxBackColorSelect = ""  
 ,textboxBorderColorSelect = "rgb(0,200, 0)"  
   
 ### tables  
 ,tableBackColor = "rgb(240, 255, 255)"  
 ,tableBorderColor =""  
 ,tableBorderTopSize = 1  
 ,tableBorderRowSize = 1  
)  
header <- dashboardHeader(dropdownMenuOutput('menu1'),  
 dropdownMenuOutput('menu2'),  
 dropdownMenuOutput('menu3'),  
 userOutput('user'),  
 title = tags$a(tags$img(src='logo.png'),  
 href='','INFINICALS',  
 style = "color:yellow;   
 font-family:Candara;  
 font-size:28px;  
 font-weight: bold;")  
 )

## This Font Awesome icon ('gears') does not exist:  
## \* if providing a custom `html\_dependency` these `name` checks can   
## be deactivated with `verify\_fa = FALSE`

sidebar <-dashboardSidebar(  
 tags$h2(  
 sidebarMenu(  
 menuItem('Dashboard',  
 tabName='t\_item1',  
 icon=icon('tachometer-alt'),  
 selected= TRUE  
 ),  
 menuItem("Database",  
 tabName = "t\_item2",  
 icon = icon("database"),  
 menuSubItem('By Crop',  
 tabName = "t\_item21",  
 icon = shiny::icon("angle-double-right")  
 ),  
 menuSubItem('By Type',  
 tabName = "t\_item22",  
 icon = shiny::icon("angle-double-right")  
 ),  
 menuSubItem('Update',  
 tabName = "t\_item23",  
 icon = shiny::icon("angle-double-right")  
 )  
 ),  
 menuItem("Economic Variables",  
 tabName = "t\_item3",  
 icon = icon("chart-line"),  
 menuSubItem('Public Debt',  
 tabName = "t\_item31",  
 icon = shiny::icon("angle-double-right")  
 ),  
 menuSubItem('Inflation',  
 tabName = "t\_item32",  
 icon = shiny::icon("angle-double-right")  
 ),  
 menuSubItem('Exchange Rate',  
 tabName = "t\_item33",  
 icon = shiny::icon("angle-double-right")  
 )  
 ),  
 menuItem('Your data',  
 tabName='t\_item4',  
 icon = icon('file-import'),  
 menuSubItem('Import',  
 tabName = "t\_item41",  
 icon = shiny::icon("angle-double-right")  
 ),  
 menuSubItem('Visualise',  
 tabName = "t\_item42",  
 icon = shiny::icon("angle-double-right")  
 ),  
 menuSubItem('Statistical Summary',  
 tabName = "t\_item43",  
 icon = shiny::icon("angle-double-right")  
 )  
 ),  
 menuItem('Settings',  
 tabName='t\_item5',  
 icon = icon('hammer')  
 ),  
 menuItem('About',  
 tabName ='t\_item6',  
 icon= icon('address-card')  
 )  
 )  
 )  
 )  
  
body<- dashboardBody(  
 tags$head(tags$script(HTML('  
 Shiny.addCustomMessageHandler("jsCode",  
 function(message) {  
 console.log(message)  
 eval(message.code);  
 }  
 );  
 ')  
 )),  
 useShinyFeedback(),   
 useShinyjs(),  
 tags$head(includeScript('returnClick.js')),customTheme,  
 tags$head(tags$link(rel='stylesheet',type='text/css',  
 href='styles.css')  
 ),  
 tabItems(  
 tabItem(tabName = 't\_item1',  
 fluidRow(  
 column(12,  
 titlePanel(tags$h3('DATA FOR A SINGLE CROP BASED ON TYPE'))  
 )  
 )  
 ),  
 tabItem(tabName = "t\_item21",  
 fluidRow(  
 column(12,  
 titlePanel(tags$h3('COMPANY DATA')),  
 tabsetPanel(  
 tabPanel(title='Data Table',  
 icon=icon('table'),  
 titlePanel('Ranked from latest date of record'),br(),  
 sidebarLayout(  
 sidebarPanel(  
 tags$style(".well {background-color: #f0ffff;}"),  
 selectInput(inputId="col\_view",   
 label="Select Crops to view",  
 multiple = TRUE,  
 selected= unique(exports$dates),  
 choices = unique(colnames(exports))  
 ),  
 dateRangeInput('datum',strong('Period'),  
 start= min(exports $dates),  
 end=max(exports $dates),  
 min=min(exports$dates),  
 max=max(exports$dates)  
 )  
 ),  
 mainPanel(  
 tags$head(tags$style("table { background-color: #ffff97; }", media="screen", type="text/css")),  
   
 tags$head(  
 tags$style(  
 HTML(".shiny-output-error-validation{color: #ff0000;font-weight: bold;}"  
 ))),  
 textOutput('DateRange'),  
 DT::DTOutput('table2'  
 )  
 )  
 )  
 ),  
 tabPanel(title='Visualise',  
 icon=icon('eye'),  
 titlePanel('Graphs'),  
 sidebarLayout(  
 sidebarPanel(  
 pickerInput(  
 inputId = 'select\_crop',  
 label = strong('Select Crop'),  
 multiple = TRUE,  
 options=list(`max-options`=3),  
 selected ='', choices =unique(exports1$crop)  
 ),  
 dateRangeInput(inputId = 'date2',  
 label = strong('Period'),  
 start= min(exports1$dates),  
 end=max(exports1$dates),  
 min=min(exports1$dates),  
 max=max(exports1$dates)  
 )  
 ),  
 mainPanel(  
 tags$head(  
 tags$style(  
 HTML(".shiny-output-error-validation {color: #ff0000;font-weight: bold;}"  
 ))),  
 tabsetPanel(  
 tabPanel(  
 title='Table',   
 br(),  
 textOutput('DateRange2'),  
 DT::DTOutput('table22')  
 ),  
 tabPanel(  
 title='Graphs',  
 titlePanel('Type of Graph'),  
 radioButtons(inputId='graph\_type',  
 label='Choose the Graph',  
 choices= c('Line','Normal Bar','Comparative Bar','Cumulative Bar','Pie'),  
 inline = TRUE ,   
 selected=NULL),  
 conditionalPanel(  
 condition= "input.graph\_type=='Pie'",  
 dateInput(inputId = 'date5',  
 label = strong('Pie Date'),  
 min=min(exports1$dates),  
 max=max(exports1$dates),  
 value ='2000-12-01'  
 )  
 ),  
 textOutput('text2'),  
 conditionalPanel(  
 condition ="input.graph\_type!=  
 'Cumulative Bar'",  
 withSpinner(  
 echarts4rOutput('graph2'),   
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE)  
 ),  
 conditionalPanel(  
 condition= "input.graph\_type=='Pie'",  
 column(12,  
 tags$div(id="test1",   
 style="width:100%;height:300px;"),   
 deliverChart(div\_id = "test1")  
 )  
 ),  
 conditionalPanel(  
 condition= "input.graph\_type=='Cumulative Bar'",  
 fluidRow(  
 column(12,  
 tags$div(id="test",   
 style="width:100%;height:400px;"),  
 deliverChart(div\_id = "test")  
 ))  
 )  
 )  
 )  
 )  
 )  
 ),  
 tabPanel(  
 title='Visuals page',  
 icon=icon('eye'),  
 titlePanel('Different Graphs'),  
 fluidRow(  
 column(12,  
 box(  
 solidHeader = FALSE,  
 title = "Control",  
 status = "primary",  
 pickerInput(  
 inputId = 'select\_crop1',  
 label = strong('Select Crop'),  
 multiple = TRUE,  
 options=list(`max-options`=3),  
 selected ='',   
 choices =unique(exports1$crop)  
 ),  
 dateRangeInput(inputId = 'date21',  
 label = strong('Period'),  
 start= min(exports1$dates),  
 end=max(exports1$dates),  
 min=min(exports1$dates),  
 max=max(exports1$dates)  
 ),  
 dateInput(inputId = 'date51',  
 label = strong(' % Pie Date'),  
 min=min(exports1$dates),  
 max=max(exports1$dates),  
 value ='2000-12-01'  
 )  
 )  
 )  
 ),  
 fluidRow(  
 column(width = 6,   
 withSpinner(  
 echarts4rOutput('grapha'),   
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE)  
 ),  
 column(width = 6,   
 withSpinner(  
 echarts4rOutput('graphb'),   
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE)  
 )   
 ),  
 fluidRow(  
 column(width = 6,   
 withSpinner(  
 echarts4rOutput('graphc'),   
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE)  
 ),  
 column(width = 6,   
 tags$div(id="test11",   
 style="width:100%;height:300px;"),  
 deliverChart(div\_id = "test11")  
 )  
 ),  
 fluidRow(  
 column(width = 6,   
 withSpinner(  
 echarts4rOutput('graphd'),   
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE)  
 ),  
 column(width = 6,   
 tags$div(id="test12",   
 style="width:100%;height:400px;"),  
 deliverChart(div\_id = "test12"))  
 )  
 )  
 )  
 )  
 )  
 ),  
 tabItem(tabName ='t\_item22',  
 fluidRow(  
 column(12,  
 titlePanel(tags$h3('COMPANY DATA')),  
 tabsetPanel(  
 tabPanel(title='data Table',  
 icon=icon('table'),  
 titlePanel('View by type of Crop'),  
 sidebarLayout(  
 sidebarPanel(  
 selectInput(inputId="choose\_category",  
 label= "TYPE OF EXPORTS",   
 multiple = FALSE,  
 choices = unique(exports1$type),  
 selected= unique(exports1$type)[1]  
 ),  
 selectInput(inputId="choose\_item",  
 label="Select Crop",  
 multiple= FALSE,  
 selecte='',   
 choices =c(not\_sel)   
 ),   
 dateRangeInput(inputId = 'date3',   
 label = strong('Period'),   
 start= min(exports1$dates),  
 end=max(exports1$dates),   
 min=min(exports1$dates),  
 max=max(exports1$dates)  
 ),  
 tags$head(  
 tags$style(  
 HTML(".shiny-output-error-validation {color: #ff0000;font-weight: bold;}")))  
 ),  
   
 mainPanel(  
 textOutput('text1'),  
 DT::DTOutput('tb\_chosen')  
 )  
 )  
 ),  
 tabPanel(title='Graph',  
 icon=icon('chart-area'),  
 titlePanel('A Line graph'),  
 br(),  
 box(solidHeader = FALSE,  
 title = "Period Range",  
 status = "success",  
 dateRangeInput(inputId = 'date4',  
 label = strong('Enter the Period'),  
 start= min(exports1$dates),  
 end=max(exports1$dates),  
 min=min(exports1$dates),  
 max=max(exports1$dates)  
 )  
 ),  
 br(), br(),br(), br(),br(),  
 br(),br(),br(),  
 box(solidHeader = FALSE,  
 title = "LINE GRAPH",  
 status = "success",  
 width=12,  
 tags$head(  
 tags$style(  
 HTML(".shiny-output-error-validation{color: #ff0000;font-weight: bold;}"))),  
 withSpinner(  
 echarts4rOutput('graph1'),  
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE)  
 )  
 )  
 )  
 )  
 )  
 ),  
 tabItem(tabName ='t\_item23',  
 titlePanel(tags$h3('COMPANY DATA')),  
 tabsetPanel(id='tabs',  
 tabPanel(  
 title ='View Available Data',  
 icon=icon('eye'),   
 tags$h2(paste(  
 'The available data as at',Sys.time(),'is:',sep=' ')),  
 tags$h2('1. Debt in Kenya'),tags$h2('2. Exports'),  
 tags$h2('3. inflation'),  
 tags$h2('Please note that the available data remains under review to expand the Database'),tags$h2('Any changes in this Tab effectively affects the whole application')  
 ),  
 tabPanel(title='Authenticate',  
 icon=icon('key'), titlePanel('Administrator'), actionButton('go','OPEN DATABASE'),  
 br(),br(),  
 textOutput("base")  
 ),  
 tabPanel(title ='Data',icon=icon('table'),   
 titlePanel('Add/Edit the records'),   
 selectInput(inputId='dataset',  
 label= "Select the Dataset to Update",  
 choices = dataset),  
 div(  
 class = "container",  
 div(  
 shiny::actionButton(  
 inputId = "add\_export",   
 label = "Add Export",  
 icon = shiny::icon("plus"),  
 class = "btn-success"  
 ),  
 shiny::actionButton(  
 inputId = "update",  
 label = "Update",  
 icon = shiny::icon("database"),  
 class = "btn-success"  
 )  
 )  
 ),  
 br(),  
 DT::DTOutput(outputId = "dt\_table"),  
 shiny::includeScript("script.js")  
 )  
 )  
 ),  
 tabItem(tabName = "t\_item31",  
 fluidRow(  
 column(12,  
 titlePanel(tags$h3('TREND ANALYSIS')  
 ),  
 tabsetPanel(  
 tabPanel(  
 title ='Public Debt',  
 icon = icon('coins'),  
 titlePanel('Kenya Debt'),  
 sidebarLayout(  
 sidebarPanel(  
 selectInput(inputId = "indicator1",  
 label = strong("Type of Debt"),  
 choices = unique(kenya\_status1$indicator),  
 selected = "domestic\_debt"),  
 dateRangeInput(inputId ="date",  
 label=strong("period"),  
 start =min(kenya\_status1$date),  
 end = max(kenya\_status1$date),  
 min = min(kenya\_status1$date),   
 max = max(kenya\_status1$date)  
 ),  
 checkboxInput(inputId='comp',  
 label='Compare the Trends')  
 ),  
 mainPanel(  
 tabsetPanel(  
 tabPanel('Line Graphs',  
 icon=icon('chart-line'),  
 titlePanel('Growth of Debt against time'),  
 tags$head( tags$style(  
 HTML(".shiny-output-error-validation{color: #ff0000;font-weight: bold;}")  
 )  
 ),  
 tags$a(href = "https://www.centralbank.go.ke/public-debt/", "Source: Central Bank of Kenya",  
 target = "\_blank"),  
 withSpinner(echarts4rOutput('graph3'),  
 type=1,  
 color="#b33e48",   
 hide.ui=FALSE) ),  
 tabPanel(  
 title ='Summary',  
 icon = icon('calculator'),   
 titlePanel('Constants'),  
 column(6,  
 withSpinner(echarts4rOutput('liquid'),  
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE)  
 ),  
 column(6,  
 withSpinner(echarts4rOutput('clock'),  
 type=1,   
 color="#b33e48",  
 hide.ui=FALSE)  
 )  
 )  
 )  
 )  
 )  
 )  
 )  
 )  
 )  
 ),  
 tabItem(tabName ='t\_item32',  
 titlePanel('Purchasing Power')  
 ),  
 tabItem(tabName ='t\_item33',  
 titlePanel('Shilling Strength')  
 ),  
 tabItem(tabName = 't\_item41',  
 fluidRow(  
 column(12,  
 titlePanel(tags$h3('PREPARE YOUR DATA')),  
 tabsetPanel(  
 tabPanel('Import Data',  
 icon=icon('upload'),  
 titlePanel('Upload'),  
 sidebarLayout(  
 sidebarPanel(  
 fileInput('file1','Data',buttonLabel='Upload...',accept=c('.cvs','.tsv','.xls','.xlsx')  
 ),  
 numericInput('n','No. of Rows to preview',value=10,min=1,step=1),  
 radioButtons("disp", "Display",  
 choices = c(Head = "head",  
 All = "all"),  
 selected = "head"),  
 actionButton("preview", "Preview")  
 ),  
 mainPanel(  
 tags$head(  
 tags$style(  
 HTML(".shiny-output-error-validation{color: #ff0000;font-weight: bold;}")  
 )  
 ),   
 conditionalPanel(  
 condition = "input.preview",  
 div(  
 class = "text-left",  
 div(  
 style = "display: inline-block;",  
 tags$h2("Click cell to edit"),  
 tags$h2("Left click to edit rows,columns  
 and download a csv file")  
 )  
 )  
 ),br(),  
 withSpinner(rHandsontableOutput('hot'),type=1,color="#b33e48")  
 )  
 )  
 )  
 )  
 )  
 )  
 ),  
 tabItem(tabName= 't\_item42',  
 fluidRow(  
 column(12,  
 titlePanel(tags$h3('PLOT YOUR DATA')),  
 tabsetPanel(  
 tabPanel(  
 title ='Data Plots',  
 icon = icon('cloud'),  
 titlePanel('Control your Plot'),  
 sidebarLayout(  
 sidebarPanel(  
 selectInput(inputId='axis1',  
 label= "X-Axis",  
 choices = c(not\_sel)  
 ),  
 selectInput(inputId='axis2',  
 label= "Y-Axis",  
 choices = c(not\_sel)  
 ),  
 selectInput(inputId='group',  
 label= "Group by",  
 choices = c(not\_sel)  
 ),  
 tags$h2('NB:FOR REGRESSION MODEL'),  
 checkboxInput(inputId='tick',  
 label='Line of Best fit'),  
 checkboxInput("se",   
 "Add confidence interval around the regression line",FALSE),  
 br(),  
 actionButton("run\_button", "Run Analysis",   
 icon = icon("play")  
 )  
 ),  
 mainPanel(  
 tabsetPanel(  
 tabPanel('Regular Plot',  
 icon = icon('expeditedssl'),  
 tags$h2("Choose only numeric data for axis!"),  
 br(),  
 withSpinner(  
 plotlyOutput('graph4'),   
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE)),  
 tabPanel('Regression Model',  
 icon=icon('layer-group'),  
 titlePanel('Plot'),  
 withMathJax(),  
 tags$b("Regression plot:"),  
 uiOutput("results"),  
 numericInput('predict','Predict',  
 value=0),   
 verbatimTextOutput("value1"),  
 br(),  
 withSpinner(  
 plotlyOutput('graph5'),  
 type=1,  
 color="#b33e48",  
 hide.ui=FALSE),  
 tags$b("Interpretation:"),  
 uiOutput("interpretation")  
   
 )  
 )  
   
 )  
 )))  
 )  
 )  
 ),  
 tabItem(tabName = 't\_item43',  
 titlePanel(tags$h3('STATISTICS')),  
 tags$h2('Summary Tables'),  
 fluidRow(  
 column(width = 4, strong(textOutput("num\_var\_1\_title"))  
 ),  
 column(width = 4, strong(textOutput("num\_var\_2\_title"))  
 ),  
 column(width = 4, strong(textOutput("fact\_var\_title"))  
 )  
 ),  
 fluidRow(  
 column(width = 4, tableOutput("num\_var\_1\_summary\_table")),  
 column(width = 4, tableOutput("num\_var\_2\_summary\_table")),  
 column(width = 4, tableOutput("fact\_var\_summary\_table"))  
 ),  
 fluidRow(  
 column(width = 6, strong("Combined Statistics")),  
 column(width = 6, strong("Regression Summary"))  
 ),  
 fluidRow(  
 column(width = 6, tableOutput("combined\_summary\_table")),  
 column(width = 6, tableOutput("regression\_summary\_table"))  
 )  
 ),  
 tabItem(tabName='t\_item5',  
 fluidRow(  
 column(12  
 )  
 )  
 )  
 )  
)  
ui <-tabsetPanel(  
 id = "panels",  
 type = "hidden",  
 selected = "landing\_page",  
 tabPanelBody(value = "landing\_page",  
 tags$div(  
 class = "landing\_page\_container",   
 tags$div(  
 class = "landing\_page\_header",  
 tags$div(  
 class = "landing\_page\_logo",  
 tags$a(  
 tags$img(src='logo.png')  
 ),  
 style= "float:left;"  
 ),  
 tags$div(  
 class = "select\_something",   
 actionLink('portal',  
 'USER PORTAL',  
 icon = icon("users")  
 )  
 )  
 ),  
 tags$div(  
 class = "landing\_page\_body",   
 tags$div(  
 class = "landing\_page\_content",  
 tags$div(  
 class = "text\_content",   
 tags$div(  
 class ="some\_text",  
 tags$h1(  
 tags$strong(  
 "INFINICALS",   
 style = "font-size: 200%")  
 )  
 ),   
 tags$div(  
 class ="more\_text",   
 tags$h2(  
 "Beyond Infinity",  
 style =  
 "color:#FFFFFF;")  
 )  
 )  
 )  
 )  
 )  
 ),  
 tabPanelBody(  
 value = "login\_page",  
 tags$div(id = "div\_login",  
 fluidRow(  
 align= 'center',  
 tags$div (class= 'login',  
 box(align='left',   
 title = tags$a("SIGN IN TO INFINICALS",  
 style = "font-size:20px;   
 font-weight: bold;"),  
 status="warning",  
 solidHeader= TRUE,   
 textInput(  
 "userName",  
 label = div(icon("user-plus",  
 style = "color:#c83c00;"),  
 'Username')  
 ),  
 passwordInput(  
 'passwd',  
 label = div(icon("key",  
 style ="color:#c83c00;"),  
 "Password")  
 ),  
 actionButton("Login", "Log in"),  
 footer= div(class= 'pull-right-container',  
 tagList(  
 tags$a(href='http://company.fr/',  
 'Forgotten Password',  
 style = "color:#1db3ff;"),  
 br(),  
 tags$a(href='http://jeff.com/',  
 'New User',  
 style = "color:#1db3ff;")  
 )  
 ),  
 width =4  
 )  
 )  
 )  
 )  
 ),  
 tabPanelBody(  
 value = "dashboard",  
 tags$div(id = "div\_dashboard",  
 dashboardPage(  
 header = header,   
 sidebar = sidebar,   
 body = body,  
 options = list(sidebarExpandOnHover = TRUE)  
 )  
 )  
 )  
 )  
   
 login\_details <- data.frame(user = c("JEFFERSON","SAM", "PAM", "RON"),  
 pswd = c("123A","123B","123C","123D")  
 )  
   
 draw\_plot<- function(upload, axis1, axis2, group){  
 if(group!=not\_sel){  
 upload[,(group):= as.factor(upload[,get(group)])]  
 }  
 if(axis1 == not\_sel & axis2 == not\_sel & group== not\_sel){  
 return(NULL)  
 }  
 else if(axis1 != not\_sel & axis2 != not\_sel & group!= not\_sel){  
 ggplot(data = upload,  
 aes\_string(x = axis1, y = axis2, color = group)) +  
 geom\_point()+ theme1  
 }  
 else if(axis1 != not\_sel & axis2 != not\_sel & group == not\_sel){  
 ggplot(data = upload,  
 aes\_string(x = axis1, y = axis2)) +  
 geom\_point()+theme1  
 }  
 else if(axis1!= not\_sel & axis2 == not\_sel & group != not\_sel){  
 ggplot(data = upload,  
 aes\_string(x = group, y = axis1)) +  
 geom\_violin()+theme1  
 }  
 else if(axis1 == not\_sel & axis2 != not\_sel & group != not\_sel){  
 ggplot(data = upload,  
 aes\_string(x = group, y = axis2)) +  
 geom\_violin()+theme1  
 }  
 else if(axis1!= not\_sel & axis2 == not\_sel & group == not\_sel){  
 ggplot(data = upload,  
 aes\_string(x = axis1)) +  
 geom\_histogram()+theme1  
 }  
 else if(axis1 == not\_sel & axis2 != not\_sel & group== not\_sel){  
 ggplot(data = upload,  
 aes\_string(x = axis2)) +  
 geom\_histogram()+theme1  
 }  
 else if(axis1 == not\_sel & axis2 == not\_sel & group != not\_sel){  
 ggplot(data = upload,  
 aes\_string(x = group)) +  
 geom\_bar()+theme1  
 }  
 }  
 create\_num\_var\_table <- function(upload, num\_var){  
 if(num\_var != not\_sel){  
 col <- upload[,get(num\_var)]  
 if (length(col)>5000) col\_norm <- sample(col,5000) else col\_norm <- col  
 norm\_test <- shapiro.test(col\_norm)  
 statistic <- c("mean", "median", "5th percentile", "95th percentile",  
 "Shapiro statistic", "Shapiro p-value")  
 value <- c(round(mean(col),2), round(median(col),2),  
 round(quantile(col, 0.05),2), round(quantile(col, 0.95),2),  
 norm\_test$statistic, norm\_test$p.value)  
 data.table(statistic, value)  
 }else{  
 return()  
 }  
 }  
 create\_fact\_var\_table <- function(upload, group){  
 if(group != not\_sel){  
 freq\_tbl <- upload[,.N, by = get(group)]  
 freq\_tbl <- setnames(freq\_tbl,c("factor\_value", "count"))  
 freq\_tbl  
 }else{  
 return()  
 }  
 }  
 create\_combined\_table <- function(upload, axis1, axis2, group){  
 if(group != not\_sel){  
 if(axis1 != not\_sel & axis2 != not\_sel){  
 res\_tbl <- upload[,.(correlation = cor(get(axis1), get(axis2))), by = group]  
 }  
 else if(axis1 != not\_sel & axis2 == not\_sel){  
 res\_tbl <- upload[,.(mean = mean(get(axis1))), by = group]  
 }  
 else if(axis1 == not\_sel & axis2 != not\_sel){  
 res\_tbl <- upload[,.(mean = mean(get(axis2))), by = group]  
 }  
 else if(axis1 == not\_sel & axis2 == not\_sel){  
 res\_tbl <- return()  
 }  
 }  
 else if(axis1 != not\_sel & axis2 != not\_sel){  
 res\_tbl <- data.table(  
 statistic = c("correlation"),  
 value = c(cor(  
 upload[,get(axis1)],  
 upload[,get(axis2)])))  
 }  
 else{return()}  
 return(res\_tbl)  
 }  
 create\_regression\_table <- function(upload, axis1, axis2){  
 if(axis1 != not\_sel & axis2 != not\_sel){  
 x<- upload[,get(axis1)]  
 y <- upload[,get(axis2)]  
 fit <- lm(y~x)  
 res\_tbl <- data.table(  
 parameter =c("Formular","Intercept",paste(axis2, "Gradient"),  
 'Adj. R-Squared','P-Value'),  
 value = c(paste(axis2 ,'~', axis1),  
 round(fit$coef[[1]], 3),  
 round(fit$coef[[2]], 3),  
 round(summary(fit)$adj.r.squared, 3),  
 signif(summary(fit)$coef[2, 4], 3)  
 ))  
 return(res\_tbl)  
 } else{  
 return()  
 }  
 }  
 create\_btns <- function(x) {  
 x %>%  
 purrr::map\_chr(~  
 paste0(  
 '<div class = "btn-group">  
 <button class="btn btn-default action-button btn-info action\_button" id="edit\_',  
 .x, '" type="button" onclick=get\_id(this.id)><i class="fas fa-edit"></i></button>  
 <button class="btn btn-default action-button btn-danger action\_button" id="delete\_',  
 .x, '" type="button" onclick=get\_id(this.id)><i class="fa fa-trash-alt"></i></button></div>'  
 ))  
 }  
 x <- create\_btns(1:nrow(exports1))  
 y <- create\_btns(1:nrow(kenya\_status1))  
 mtcars <- exports1%>%  
 dplyr::bind\_cols(tibble("Buttons" = x))  
 mtcars1<-kenya\_status1%>%  
 dplyr::bind\_cols(tibble("Buttons" = y))  
   
 server <- function(input, output, session) {  
   
 debt <- shiny::reactiveFileReader(1000,session,'~/Programming/R/DATA/kenya\_status1.xlsx',readFunc = function(filePath){   
 openxlsx::read.xlsx(filePath,detectDates = TRUE)  
 })  
 expos <- shiny::reactiveFileReader(1000,session, filePath = '~/Programming/R/DATA/exports1.xlsx',readFunc = function(filePath){   
 openxlsx::read.xlsx(filePath,detectDates = TRUE)  
 })  
 expot <- shiny::reactiveFileReader(1000,session,'~/Programming/R/DATA/exports.xlsx',readFunc = function(filePath){   
 openxlsx::read.xlsx(filePath,detectDates = TRUE)  
 })  
   
 observe({  
 updateDateRangeInput(session, "date",  
 start = min(debt()$date),  
 end = max(debt()$date),  
 min = min(debt()$date),  
 max = max(debt()$date)  
 )  
 })  
 observe({  
 updateDateRangeInput(session, "date2",  
 start = min(expos()$dates),  
 end = max(expos()$dates),  
 min = min(expos()$dates),  
 max = max(expos()$dates)  
 )  
 })  
 observe({  
 updateDateRangeInput(session, "date3",  
 start = min(expos()$dates),  
 end = max(expos()$dates),  
 min = min(expos()$dates),  
 max = max(expos()$dates)  
 )  
 })  
 observe({  
 updateDateRangeInput(session, "date4",  
 start = min(expos()$dates),  
 end = max(expos()$dates),  
 min = min(expos()$dates),  
 max = max(expos()$dates)  
 )  
 })  
 observe({  
 updateDateRangeInput(session, "datum",  
 start = min(expot()$dates),  
 end = max(expot()$dates),  
 min = min(expot()$dates),  
 max = max(expot()$dates)  
 )  
 })  
   
 login.page =   
 paste(  
 isolate(session$clientData$url\_protocol),  
 "//",  
 isolate(session$clientData$url\_hostname),  
 ":",  
 isolate(session$clientData$url\_port),  
 sep = ""  
 )  
 USER <- reactiveValues(Logged = F)  
 observe({  
 if (USER$Logged == FALSE) {  
 if (!is.null(input$Login)) {  
 if (input$Login > 0) {  
 Username <- isolate(input$userName)  
 Password <- isolate(input$passwd)  
 if (nrow(login\_details[login\_details$user == Username &   
 login\_details$pswd == Password,]) >= 1) {  
 USER$Logged <- TRUE  
 }  
 }  
 }  
 }  
 })  
   
   
 # hide dashboard by default:  
 jqui\_hide(  
 ui = "#div\_dashboard",   
 effect = "blind"  
 )  
 #hide login\_page by default  
 jqui\_hide(  
 ui = "#div\_login",   
 effect = "blind"  
 )  
 observeEvent(input$portal,{  
 updateTabsetPanel(  
 session = session,   
 inputId = "panels",   
 selected = "login\_page"  
 )  
 #show login contents  
 jqui\_show(  
 ui = "#div\_login",   
 effect = "blind",   
 duration = 500  
 )  
 })  
   
 observeEvent(input$Login, {   
 if(USER$Logged!=TRUE){  
 shinyFeedback::showFeedbackDanger(inputId = 'passwd',  
 text = 'INVALID DETAILS!')  
 shinyFeedback::showFeedbackDanger(inputId = 'userName',text = '')  
 } else {  
 hideFeedback('passwd')  
 hideFeedback('userName')  
 shinyalert('PLEASE WAIT', "Signing in...",   
 type = "info",   
 timer = 800,  
 showConfirmButton = FALSE,  
 animation = "pop"  
 )  
 updateTabsetPanel(  
 session = session,   
 inputId = "panels",   
 selected = "dashboard"  
 )  
 jqui\_show(  
 ui = "#div\_dashboard",   
 effect = "blind",   
 duration =500  
 )  
 showNotification(ui=paste('Dear', input$userName,",Welcome to Infinicals",sep = ' '),  
 type='message',  
 closeButton=FALSE,  
 duration = 3  
 )  
 }  
 })  
   
 output$menu1 <- renderMenu({  
 dropdownMenu(type ="notifications", badgeStatus = "warning",  
 notificationItem(icon = icon("users"), status = "info",  
 "5 new members joined today!"),  
 notificationItem(icon = icon("broom"), status = "danger",  
 "scan your device for virus!"),  
 notificationItem(icon = icon("gamepad"), status = "success",  
 "You are in level 5!")  
 )  
 })  
 output$menu2<- renderMenu({  
 dropdownMenu(type='messages',badgeStatus='info',  
 messageItem(from='@JoeAgeyo',  
 message='You have been appointed the Director!',  
 icon=icon('signature'),  
 time='Yesterday'),  
 messageItem(from='@AgnesKavindu',  
 message='Your Business Proposal has been approved',  
 icon=icon('briefcase'),  
 time='just now'),  
 messageItem(from='@Dad',  
 message='Hi,Could you pass by Naivas and have some cereals for me, thanks',  
 icon=icon('user-circle'),  
 time='2022-02-27 17:09')  
   
 )  
 })  
 output$menu3 <- renderMenu({  
 dropdownMenu(type='tasks',badgeStatus='success',  
 taskItem('XET 301 Assignment',  
 value=70,  
 color='teal'),  
 taskItem('STA 222 Assignment',  
 value=45,  
 color='maroon'),  
 taskItem('STA 402 Assignment',  
 value=94,  
 color='lime'),  
 taskItem('XEA 102 Assignment',  
 value=12,  
 color='green')  
 )  
 })  
 output$user <-renderUser({  
 dashboardUser(  
 name=input$userName,  
 image='user\_image.png',  
 footer=p('Beyond Infinity',class='text-centre'),  
 subtitle = a(icon("user"),"LOG OUT", href = login.page)  
 )  
 })  
 output$text1<-renderText({  
 req(input$date3)  
 validate(need(!is.na(input$date3[1]) & !is.na(input$date3[2]), "Error: Please provide both a start and an end date."))  
 validate(need(input$date3[1] < input$date3[2], "Error: Start date should be earlier than end date."))  
 })  
   
 shiny::observeEvent(input$choose\_category, {  
 x<- c(not\_sel,unique(exports1 [  
 exports1$type%in%input$choose\_category,"crop"])  
 )  
 shiny::updateSelectInput(session,  
 'choose\_item',  
 label=paste('Select',  
 input$choose\_category,sep = ' '),  
 choices=x,  
 selected= not\_sel  
 )  
 })  
 output$tb\_chosen <- DT::renderDT (  
 expos()%>%   
 filter(  
 expos()$type%in%input$choose\_category &  
 expos()$crop%in%input$choose\_item & dates > as.POSIXct(input$date3[1]) & dates < as.POSIXct(input$date3[2]))  
 )  
 datae2<-reactive({  
 req(input$choose\_item)  
 req(input$date4)  
 validate(need(!is.na(input$date4[1]) & !is.na(input$date4[2]), "Error: Please provide both a start and an end date."))  
 validate(need(input$date4[1] < input$date4[2], "Error: Start date should be earlier than end date."))  
 expos() %>%filter(  
 expos() $type%in%input$choose\_category &  
 expos()$crop%in%input$choose\_item & dates > as.POSIXct(input$date4[1]) & dates < as.POSIXct(input$date4[2]))  
 })  
 output$graph1<-renderEcharts4r({  
 datae2()|>   
 group\_by(crop)|>  
 e\_charts(dates)|>  
 e\_line(weight,symbol='none')|>   
 e\_animation(duration = 4000)|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Dates',y = 'Weight Exported in Kgs.')|>   
 e\_title(paste('Weight of',input$choose\_item,'exported',sep=' '),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',right = '5', top = '15%')|>  
 e\_color(my\_colors)  
 })  
 output$DateRange <- renderText({  
 req(input$datum)  
 validate(need(!is.na(input$datum[1]) & !is.na(input$datum[2]),  
 "Error: Please provide both a start and an end date."))  
 validate(need(input$datum[1] < input$datum[2], "Error: Start date should be earlier than end date."))  
 })  
 output$table2<- DT::renderDT(  
 expot() %>% select(dates,input$col\_view)%>% filter(  
 dates > as.POSIXct(input$datum[1]) & dates < as.POSIXct(input$datum[2])))  
 output$DateRange2 <- renderText({  
 req(input$date2)  
 validate(need(!is.na(input$date2[1]) & !is.na(input$date2[2]), "Error: Please provide both a start and an end date."))  
 validate(need(input$date2[1] < input$date2[2], "Error: Start date should be earlier than end date."))  
 })  
 output$text2<-renderText({  
 validate(need(!is.na(input$select\_crop), "Select a maximum of three Crops"))  
 validate(need(!is.na(input$date2[1]) & !is.na(input$date2[2]), "Error: Please provide both a start and an end date."))  
 validate(need(input$date2[1] < input$date2[2], "Error: Start date should be earlier than end date."))  
 })  
 output$table22<- DT::renderDT(  
 expos() %>% select(dates,crop,weight)%>% filter(expos()$crop%in% input$select\_crop,dates > as.POSIXct(input$date2[1]) & dates < as.POSIXct(input$date2[2])))  
 datae3<-reactive({  
 validate(need(!is.na(input$select\_crop), "You have not selected any crop to view"))  
 expos() %>% select(dates,crop,weight)%>% filter(expos()$crop%in% input$select\_crop,dates > as.POSIXct(input$date2[1]) & dates < as.POSIXct(input$date2[2]))  
 })  
   
 data <- reactive({  
 validate(need(!is.na(input$select\_crop), "You have not selected any crop to view"))  
 validate(need(!is.na(input$date5), "Pie Date required\*"))  
 total <- expos()%>% filter(expos()$crop%in%input$select\_crop,expos()$dates%in%input$date5) %>% select(weight)%>% sum()  
 weight1<- expos()%>%filter(expos()$crop%in%input$select\_crop[1],expos()$dates%in%input$date5)%>%select(weight)%>%sum()  
 weight2<- expos()%>%filter(expos()$crop%in%input$select\_crop[2],expos()$dates%in%input$date5)%>%select(weight)%>%sum()  
 weight3<-expos()%>%filter(expos()$crop%in%input$select\_crop[3],expos()$dates%in%input$date5)%>%select(weight)%>%sum()  
 percent1 <- round((weight1/total)\*100,0)  
 percent2 <-round((weight2/total)\*100,0)  
 percent3 <-round((weight3/total)\*100,0)  
 data.frame(dates=c(input$date5,input$date5,input$date5),  
 crop= c(input$select\_crop[1],input$select\_crop[2],input$select\_crop[3]),  
 weight= c(percent1,percent2,percent3) )  
   
 })   
 datay <- reactive({  
 validate(need(!is.na(input$select\_crop1), "You have not selected any crop to view"))  
 validate(need(!is.na(input$date51), "Pie Date required\*"))  
 total <- expos()%>% filter(expos()$crop%in%input$select\_crop1,expos()$dates%in%input$date51) %>% select(weight)%>% sum()  
 weight1<- expos()%>%filter(expos()$crop%in%input$select\_crop1[1],expos()$dates%in%input$date51)%>%select(weight)%>%sum()  
 weight2<- expos()%>%filter(expos()$crop%in%input$select\_crop1[2],expos()$dates%in%input$date51)%>%select(weight)%>%sum()  
 weight3<-expos()%>%filter(expos()$crop%in%input$select\_crop1[3],expos()$dates%in%input$date51)%>%select(weight)%>%sum()  
 percent1 <- round((weight1/total)\*100,0)  
 percent2 <-round((weight2/total)\*100,0)  
 percent3 <-round((weight3/total)\*100,0)  
 data.frame(dates=c(input$date51,input$date51,input$date51),  
 crop= c(input$select\_crop1[1],input$select\_crop1[2],input$select\_crop1[3]),  
 weight= c(percent1,percent2,percent3) )  
   
 })   
 style <- reactive({  
 input$graph\_type  
 })  
   
 my\_colors<-c("#b33e48", "#08b582", "#4c5138 ")  
   
 plottype <- reactive({  
 switch(style(),  
 "Line" = datae3()|>   
 group\_by(crop)|>  
 e\_charts(dates)|>  
 e\_line(weight,symbol='none')|>   
 e\_animation(duration = 4000)|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Dates',y = 'Weight Exported in Kgs.')|>   
 e\_title(paste('A',input$graph\_type,'Graph of',input$select\_crop[1],',',input$select\_crop[2],'and',input$select\_crop[3]),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',right = '5', top = '15%')|>  
 e\_color(my\_colors),  
 'Normal Bar'= datae3()|>   
 group\_by(crop)|>  
 e\_charts(dates,timeline=TRUE)|>  
 e\_bar(weight)|>  
 e\_animation(duration = 4000)|>  
 e\_timeline\_opts(autoPlay = TRUE, top = "55")|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Dates',y = 'Weight Exported in Kgs.')|>   
 e\_title(paste('A',input$graph\_type,'Graph of',input$select\_crop[1],',',input$select\_crop[2],'and',input$select\_crop[3]),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',   
 right = '5', top = '15%')|>  
 e\_color(my\_colors),  
 "Comparative Bar" = datae3()|>   
 group\_by(crop)|>  
 e\_charts(dates)|>  
 e\_bar(weight)|>  
 e\_animation(duration = 4000)|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Dates',y = 'Weight Exported in Kgs.')|>   
 e\_title(paste('A',input$graph\_type,'Graph of',input$select\_crop[1],',',input$select\_crop[2],'and',input$select\_crop[3]),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',   
 right = '5', top = '15%')|>  
 e\_color(my\_colors),  
 'Pie'= data() |>  
 group\_by(dates)|>  
 e\_charts(crop)|>  
 e\_pie(weight)|>  
 e\_tooltip() |>  
 e\_animation(duration = 4000)|>  
 e\_title(paste('A pie Graph % weight of',  
 input$select\_crop[1],',',input$select\_crop[2],  
 'and',input$select\_crop[3],'on',input$date5)) |>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',   
 right = '5', top = '15%')|>  
 e\_color(my\_colors)  
 )  
 })  
 datam <- reactive({  
 req(input$date2)  
 validate(need(!is.na(input$select\_crop),  
 "You have not selected any crop"))  
 exports %>%   
 select(dates,input$select\_crop)%>%   
 filter(dates > as.POSIXct(input$date2[1]) &  
 dates < as.POSIXct(input$date2[2])) %>%  
 data.frame(row.names = 1)  
 })  
 datam1 <- reactive({  
 req(input$date21)  
 validate(need(!is.na(input$select\_crop1),  
 "You have not selected any crop"))  
 exports %>%   
 select(dates,input$select\_crop1)%>%   
 filter(dates > as.POSIXct(input$date21[1]) &  
 dates < as.POSIXct(input$date21[2])) %>%  
 data.frame(row.names = 1)  
 })  
   
 observe({  
 if(input$graph\_type=='Cumulative Bar'){  
 req(input$date2)  
 validate(need(!is.na(input$select\_crop),  
 "You have not selected any crop to view"))  
 renderBarChart(div\_id = "test",   
 grid\_left = '10%',   
 data=datam()  
 )  
 } else {  
 plottype()  
 }  
 })  
 observe({  
 req(input$date21)  
 validate(need(!is.na(input$select\_crop1),  
 "You have not selected any crop to view"))  
 renderBarChart(div\_id = "test11",   
 grid\_left = '10%', theme ='dark-digerati',  
 data=datam1()  
 )  
 })  
 output$graph2<-renderEcharts4r({  
 plottype()   
 })  
 datas <- reactive({  
 validate(need(!is.na(input$select\_crop),   
 "You have not selected any crop to view"))  
 validate(need(!is.na(input$select\_crop[1]), "You have not selected any Column"))  
 validate(need(!is.na(input$select\_crop[2]), "You have not selected any Column"))  
 validate(need(!is.na(input$select\_crop[3]), "You have not selected any Column"))  
 validate(need(!is.na(input$date2), "Pie Date required\*"))  
 total <- exports1%>% filter(exports1$crop%in%input$select\_crop,dates > as.POSIXct(input$date2[1]) & dates < as.POSIXct(input$date2[2])) %>% select(weight)%>%sum()  
 weight1 <- exports1%>% filter(exports1$crop%in%input$select\_crop[1],dates > as.POSIXct(input$date2[1]) & dates < as.POSIXct(input$date2[2])) %>% select(weight)%>%sum()  
 weight2<- exports1%>% filter(exports1$crop%in%input$select\_crop[2],dates > as.POSIXct(input$date2[1]) & dates < as.POSIXct(input$date2[2])) %>% select(weight)%>%sum()  
 weight3<-exports1%>% filter(exports1$crop%in%input$select\_crop[3],dates > as.POSIXct(input$date2[1]) & dates < as.POSIXct(input$date2[2])) %>% select(weight)%>%sum()  
 values <- c(  
 rep(input$select\_crop[1],weight1),  
 rep(input$select\_crop[2],weight2),  
 rep(input$select\_crop[3],weight3)  
 )  
 })  
 dataw <- reactive({  
 validate(need(!is.na(input$select\_crop1),   
 "You have not selected any crop to view"))  
 validate(need(!is.na(input$select\_crop1[1]), "You have not selected any Column"))  
 validate(need(!is.na(input$select\_crop1[2]), "You have not selected any Column"))  
 validate(need(!is.na(input$select\_crop1[3]), "You have not selected any Column"))  
 validate(need(!is.na(input$date21), "Pie Date required\*"))  
 total <- exports1%>% filter(exports1$crop%in%input$select\_crop1,dates > as.POSIXct(input$date21[1]) & dates < as.POSIXct(input$date21[2])) %>% select(weight)%>%sum()  
 weight1 <- exports1%>% filter(exports1$crop%in%input$select\_crop1[1],dates > as.POSIXct(input$date21[1]) & dates < as.POSIXct(input$date21[2])) %>% select(weight)%>%sum()  
 weight2<- exports1%>% filter(exports1$crop%in%input$select\_crop1[2],dates > as.POSIXct(input$date21[1]) & dates < as.POSIXct(input$date21[2])) %>% select(weight)%>%sum()  
 weight3<-exports1%>% filter(exports1$crop%in%input$select\_crop1[3],dates > as.POSIXct(input$date21[1]) & dates < as.POSIXct(input$date21[2])) %>% select(weight)%>%sum()  
 values <- c(  
 rep(input$select\_crop1[1],weight1),  
 rep(input$select\_crop1[2],weight2),  
 rep(input$select\_crop1[3],weight3)  
 )  
 })  
 observe({  
 req(input$date2[1])  
 req(input$date2[2])  
 req(input$select\_crop[1])  
 req(input$select\_crop[2])  
 req(input$select\_crop[3])  
 renderPieChart(div\_id = "test1",  
 data = datas(),  
 radius = "70%",center\_x = "50%", center\_y = "50%")  
 })  
 observe({  
 req(input$date21[1])  
 req(input$date21[2])  
 req(input$select\_crop1[1])  
 req(input$select\_crop1[2])  
 req(input$select\_crop1[3])  
 renderPieChart(div\_id = "test12",  
 data = dataw(),theme ='dark-digerati',  
 radius = "70%",center\_x = "50%", center\_y = "50%")  
 })  
 datae <-reactive({  
 req(input$date)  
 debt() %>%  
 filter(  
 indicator%in%input$indicator1,  
 date > as.POSIXct(input$date[1]) & date < as.POSIXct(input$date[2])  
 )  
 })  
 datae4<-reactive({  
 req(input$date)  
 debt() %>%  
 filter(  
 date > as.POSIXct(input$date[1]) & date < as.POSIXct(input$date[2])  
 )  
 })  
   
 output$graph3<-renderEcharts4r({  
 validate(need(!is.na(input$date[1]) & !is.na(input$date[2]), "Error: Please provide both a start and an end date."))  
 validate(need(input$date[1] < input$date[2], "Error: Start date should be earlier than end date."))  
 if(input$comp==TRUE){  
 datae4()|>   
 group\_by(indicator)|>  
 e\_charts(date)|>  
 e\_line(value,symbol='none')|>   
 e\_animation(duration = 4000)|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Years',y = 'Level of debt in Ksh.')|>  
 e\_title(paste('A Comparative Graph of Domestic and External Debt in Kenya'),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',right = '5', top = '15%')|>  
 e\_color(my\_colors)  
 }else{  
 datae()|>  
 e\_charts(date)|>  
 e\_line(value,symbol='none')|>  
 e\_animation(duration = 4000)|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Years',y = 'Level of debt in Ksh.')|>  
 e\_title(paste('Trend in',input$indicator1,'in Kenya'),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_color(my\_colors)|>  
 e\_legend(orient = 'vertical',   
 right = '5', top = '15%')  
 }  
 })  
 dataz<-reactive({  
 validate(need(!is.na(input$select\_crop1), "You have not selected any crop to view"))  
 expos() %>% select(dates,crop,weight)%>% filter(expos()$crop%in% input$select\_crop1,dates > as.POSIXct(input$date21[1]) & dates < as.POSIXct(input$date21[2]))  
 })  
 output$grapha <- renderEcharts4r({  
 dataz()|>   
 group\_by(crop)|>  
 e\_charts(dates)|>  
 e\_line(weight,symbol='none')|>   
 e\_animation(duration = 4000)|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Dates',y = 'Weight Exported in Kgs.')|>   
 e\_title(paste('A Line','Graph of',input$select\_crop1[1],',',input$select\_crop1[2],'and',input$select\_crop1[3]),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',right = '5', top = '15%')|>  
 e\_color(my\_colors)|> e\_theme("dark-mushroom")  
 })  
 output$graphb <- renderEcharts4r({  
 dataz()|>   
 group\_by(crop)|>  
 e\_charts(dates,timeline=TRUE)|>  
 e\_bar(weight)|>  
 e\_animation(duration = 4000)|>  
 e\_timeline\_opts(autoPlay = TRUE, top = "55")|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Dates',y = 'Weight Exported in Kgs.')|>   
 e\_title(paste('A Bar Graph of',input$select\_crop1[1],',',input$select\_crop1[2],'and',input$select\_crop1[3]),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',   
 right = '5', top = '15%')|>  
 e\_color(my\_colors)|> e\_theme("dark-mushroom")  
 })  
 output$graphc <- renderEcharts4r({  
 dataz()|>   
 group\_by(crop)|>  
 e\_charts(dates)|>  
 e\_bar(weight)|>  
 e\_animation(duration = 4000)|>  
 e\_tooltip(trigger='axis')|>  
 e\_axis\_labels(x='Dates',y = 'Weight Exported in Kgs.')|>   
 e\_title(paste('A Comparative Bar Graph of',input$select\_crop1[1],',',input$select\_crop1[2],'and',input$select\_crop1[3]),  
 left='center',top=10)|>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',   
 right = '5', top = '15%')|>  
 e\_color(my\_colors)|> e\_theme("dark-mushroom")  
 })  
 output$graphd <- renderEcharts4r({  
 datay() |>  
 group\_by(dates)|>  
 e\_charts(crop)|>  
 e\_pie(weight)|>  
 e\_tooltip() |>  
 e\_animation(duration = 4000)|>  
 e\_title(paste('A Pie Graph % weight on',  
 input$date51)) |>  
 e\_toolbox\_feature(feature = "saveAsImage")|>  
 e\_legend(orient = 'vertical',   
 right = '5', top = '15%')|>  
 e\_color(my\_colors)|> e\_theme("dark-mushroom")  
 })  
   
   
 doc<- data.frame(val = c(0.9, 0.5, 0.4))  
 output$liquid<- renderEcharts4r({  
 doc|>  
 e\_charts()|>  
 e\_liquid(val)|>  
 e\_animation(duration = 4000)|>  
 e\_title('Total Debt to Debt Ceiling Ratio ',left='center')  
 })  
 output$clock<-renderEcharts4r({  
 e\_charts() |>   
 e\_gauge(62, "PERCENT") |>   
 e\_animation(duration = 4000)|>  
 e\_title('Total Debt to GDP Ratio',left='center')  
 })  
   
 options(shiny.maxRequestSize=10\*1024^2)   
   
 upload <- reactive({  
 req(input$file1)  
 myfile <- input$file1  
 ext <- tools::file\_ext(myfile$name)  
 datas <-switch(ext,  
 csv = read.csv(file =myfile$datapath),  
 tsv = read.tsv(file=myfile$datapath),  
 xls = readxl::read\_xls(path=myfile$datapath),  
 xlsx = readxl::read\_xlsx(path=myfile$datapath),  
 validate("Invalid file: Please upload a .csv , .tsv , .xls or .xlsx file!")  
 )  
 data.table::as.data.table(datas)  
 })  
   
 df1 <- eventReactive(input$preview,{  
 validate(need(input$n< 11,"Error: Preview a Max of 10 rows!"))  
 validate(need(input$n > 1,"Error: Preview from 1 row!"))  
 validate(need(!is.na(input$file1), "Error: You have not uploaded any file!"))  
 validate(need(!is.na(input$n), "Error: Number of rows can't be empty!"))  
 withProgress(message = 'Previewing...', value = 0, {  
 n<- input$n  
 for (i in 1:n) {  
 incProgress(1/n, detail = paste("Loading",(i/n)\*100,'%'))  
 Sys.sleep(0.04)  
 }  
 })  
 if(input$disp == "head") {  
 return(head(upload(),input$n))  
 }  
 else {  
 return(upload())  
 }  
 })  
 observeEvent(input$n, {  
 shinyFeedback::feedbackWarning('n',input$n > 10,"\*Exceeded limit")  
 shinyFeedback::feedbackWarning('n',input$n < 1,"\*Exceeded limit")  
 })  
 output$hot <- rhandsontable::renderRHandsontable (  
 data <- rhandsontable::rhandsontable(df1(),width = 700, height = 610,  
 stretchH = "all",useTypes=FALSE)%>%  
 hot\_table(highlightCol = TRUE, highlightRow = TRUE)%>%  
 hot\_context\_menu(  
 customOpts = list(  
 csv = list(name = "Download to CSV",  
 callback = htmlwidgets::JS(  
 "function (key, options) {  
 var csv = csvString(this, sep=',', dec='.');  
  
 var link = document.createElement('a');  
 link.setAttribute('href', 'data:text/plain;charset=utf-8,' +  
 encodeURIComponent(csv));  
 link.setAttribute('download', 'data.csv');  
  
 document.body.appendChild(link);  
 link.click();  
 document.body.removeChild(link);  
 }"))))  
 )  
 observeEvent(upload(),{  
 showNotification(ui="A comma-deliminated csv data is recommended for better analysis",  
 type='message',  
 duration= 10  
 )  
 choices <- c(not\_sel,names(upload()))  
   
 updateSelectInput(session, "axis1", choices = choices)  
 updateSelectInput(session, "axis2", choices = choices)  
 updateSelectInput(session, "group", choices = choices)  
 })  
 axis1 <- eventReactive(input$run\_button,input$axis1)  
 axis2 <- eventReactive(input$run\_button,input$axis2)  
 group <- eventReactive(input$run\_button,input$group)  
 graphty<- eventReactive(input$run\_button,{  
 validate(need(!is.na(input$file1), "Error: You have not uploaded any file!"))  
 draw\_plot(upload(), axis1(),axis2(), group())  
 })  
 graphty2<- eventReactive(input$run\_button,{  
 validate(need(!is.na(input$file1), "Waiting for an Upload to load..."))  
 if(input$axis1 !=not\_sel & input$axis2 !=not\_sel){  
 k<- ggplot(data = upload(),  
 aes\_string(x = input$axis1, y = input$axis2)) +  
 geom\_point()+ theme1  
 if (input$se==TRUE & input$tick==TRUE){  
 k+stat\_smooth(method='lm', col='yellow',se=TRUE,size=1,linetype=1)  
 }  
 else if(input$tick==TRUE){  
 k+ stat\_smooth(method='lm',col='yellow',se=FALSE,size=1,linetype=1)  
 }  
 else if(input$se==TRUE){  
 k+stat\_smooth(method='lm',se=input$se, col='yellow',size=1,linetype=1)  
 }  
 else {  
 return(k)  
 }  
 }else {  
 return()  
 }  
 })  
 output$graph4 <- renderPlotly(graphty())  
 output$results <- renderUI({  
 if(axis1() !=not\_sel & axis2() !=not\_sel){  
 x <- upload()[,get(axis1())]  
 y <- upload()[,get(axis2())]  
 fit<- lm(y ~ x)   
 withMathJax(  
 br(),  
 paste0("\\( \\Rightarrow y = \\hat{\\beta}\_0 + \\hat{\\beta}\_1 x = \\) ", round(fit$coef[[1]], 3), " + ", round(fit$coef[[2]], 3), "\\( x \\)")  
 )} else { return()}  
 })  
 output$graph5 <- renderPlotly(graphty2())  
 output$num\_var\_1\_title <- renderText(paste("X Variable:",axis1()))  
 num\_var\_1\_summary\_table <- eventReactive(input$run\_button,{  
 create\_num\_var\_table(upload(), axis1())  
 })  
 output$num\_var\_1\_summary\_table <- renderTable({  
 num\_var\_1\_summary\_table() })  
   
 output$num\_var\_2\_title <- renderText(paste("Y Variable:",axis2()))  
   
 num\_var\_2\_summary\_table <- eventReactive(input$run\_button,{  
 create\_num\_var\_table(upload(), axis2())  
 })  
   
 output$num\_var\_2\_summary\_table <- renderTable({  
 num\_var\_2\_summary\_table() })  
   
 output$fact\_var\_title <- renderText(paste("Factor Variable:",group()))  
   
 fact\_var\_summary\_table <- eventReactive(input$run\_button,{  
 create\_fact\_var\_table(upload(), group())  
 })  
   
 output$fact\_var\_summary\_table <- renderTable(  
 fact\_var\_summary\_table(),colnames = FALSE)  
 combined\_summary\_table <- eventReactive(input$run\_button,{  
 create\_combined\_table(upload(), axis1(), axis2(), group())  
 })  
   
 output$combined\_summary\_table <- renderTable({  
 combined\_summary\_table()  
 })  
 regression\_table <- eventReactive(input$run\_button, {  
 create\_regression\_table(upload(), axis1(), axis2())  
 })  
 output$regression\_summary\_table <- renderTable({  
 regression\_table()  
 })  
 output$interpretation <- renderUI({  
 if(axis1() !=not\_sel & axis2() !=not\_sel){  
 x <- upload()[,get(axis1())]  
 y <- upload()[,get(axis2())]  
 fit<- lm(y ~ x)   
 if (summary(fit)$coefficients[1, 4] < 0.05 & summary(fit)$coefficients[2, 4] < 0.05) {  
 withMathJax(  
 paste0("(Make sure the assumptions for linear regression (independance, linearity, normality and homoscedasticity) are met before interpreting the coefficients.)"),  
 br(),  
 paste0("For a (hypothetical) value of ", input$axis1, " = 0, the mean of ", input$axis2, " = ", round(fit$coef[[1]], 3), "."),  
 br(),  
 paste0("For an increase of one unit of ", input$axis1, ", ", input$axis2, ifelse(round(fit$coef[[2]], 3) >= 0, " increases (on average) by ", " decreases (on average) by "), abs(round(fit$coef[[2]], 3)), ifelse(abs(round(fit$coef[[2]], 3)) >= 2, " units", " unit"), ".")  
 )  
 } else if (summary(fit)$coefficients[1, 4] < 0.05 & summary(fit)$coefficients[2, 4] >= 0.05) {  
 withMathJax(  
 paste0("(Make sure the assumptions for linear regression (independance, linearity, normality and homoscedasticity) are met before interpreting the coefficients.)"),  
 br(),  
 paste0("For a (hypothetical) value of ", input$axis1, " = 0, the mean of ", input$axis2, " = ", round(fit$coef[[1]], 3), "."),  
 br(),  
 paste0("\\( \\beta\_1 \\)", " is not significantly different from 0 (p-value = ", round(summary(fit)$coefficients[2, 4], 3), ") so there is no significant relationship between ", input$axis1, " and ", input$axis2, ".")  
 )  
 } else if (summary(fit)$coefficients[1, 4] >= 0.05 & summary(fit)$coefficients[2, 4] < 0.05) {  
 withMathJax(  
 paste0("(Make sure the assumptions for linear regression (independance, linearity, normality and homoscedasticity) are met before interpreting the coefficients.)"),  
 br(),  
 paste0("\\( \\beta\_0 \\)", " is not significantly different from 0 (p-value = ", round(summary(fit)$coefficients[1, 4], 3), ") so when ", input$axis1, " = 0, the mean of ", input$axis2, " is not significantly different from 0."),  
 br(),  
 paste0("For an increase of one unit of ", input$axis1, ", ", input$axis2, ifelse(round(fit$coef[[2]], 3) >= 0, " increases (on average) by ", " decreases (on average) by "), abs(round(fit$coef[[2]], 3)), ifelse(abs(round(fit$coef[[2]], 3)) >= 2, " units", " unit"), ".")  
 )  
 } else {  
 withMathJax(  
 paste0("(Make sure the assumptions for linear regression (independance, linearity, normality and homoscedasticity) are met before interpreting the coefficients.)"),  
 br(),  
 paste0("\\( \\beta\_0 \\)", " and ", "\\( \\beta\_1 \\)", " are not significantly different from 0 (p-values = ", round(summary(fit)$coefficients[1, 4], 3), " and ", round(summary(fit)$coefficients[2, 4], 3), ", respectively) so the mean of ", input$axis2, " is not significantly different from 0.")  
 )  
 }  
 }else{  
 return()  
 }  
 })  
 output$value1 <- renderText({  
 if(axis1() !=not\_sel & axis2() !=not\_sel){  
 x <- upload()[,get(axis1())]  
 y <- upload()[,get(axis2())]  
 fit<- lm(y ~ x)   
 value <- round(fit$coef[[1]], 3)+ round(fit$coef[[2]], 3)\*input$predict  
 text<- paste('The predicted(approximated)', axis2() , 'at value', input$predict, 'of',axis1(), 'is', value )  
 return(text)  
 } else {   
 return()  
 }  
 })  
 observeEvent(input$dataset,{  
 if(input$dataset%in%'Exports'){  
 updateActionButton(session, 'add\_export',label='Add Export')  
 output$dt\_table <- renderDT(  
 {  
 shiny::isolate(rv$df)  
 },  
 escape = F,  
 rownames = FALSE,  
 options = list(processing = FALSE)  
 )  
 }else if(input$dataset%in% 'Kenya Debt'){  
 updateActionButton(session, 'add\_export',label='Add Debt Value')  
 output$dt\_table <- renderDT(  
 {  
 shiny::isolate(rv1$df)  
 },  
 escape = F,  
 rownames = FALSE,  
 options = list(processing = FALSE)  
 )  
 }else if(input$dataset%in% 'Remittances'){  
 updateActionButton(session, 'add\_export',label='Add Remittances Records')  
 }else{  
 return()  
 }  
 })  
 shiny::observe({  
 x<- unique(exports1[exports1$type%in%input$type,"crop"])  
 shiny::updateSelectInput(session,  
 'crop',  
 label=paste('Select',input$type,sep = ' '),  
 choices=x,  
 selected=x[1]  
 )  
 })  
 shiny::observeEvent(input$add\_export, {  
 if(input$dataset%in%'Exports'){  
 modal\_dialog(  
 dates = "", type = "", crop = "", weight= "", edit = FALSE  
 )  
 rv$add\_or\_edit <- 1  
 }else if(input$dataset%in%'Kenya Debt'){  
 modal\_dialog1(  
 date = "", indicators = "",value = "", edit = FALSE  
 )  
 rv1$add\_or\_edit <- 1  
 }else{  
 return()  
 }  
 })  
 rv <- shiny::reactiveValues(  
 df = mtcars,  
 dt\_row = NULL,  
 add\_or\_edit = NULL,  
 edit\_button = NULL,  
 keep\_track\_id = nrow(mtcars) + 1  
 )  
 rv1 <- shiny::reactiveValues(  
 df = mtcars1,  
 dt\_row = NULL,  
 add\_or\_edit = NULL,  
 edit\_button = NULL,  
 keep\_track\_id = nrow(mtcars1) + 1  
 )  
 shiny::observeEvent(input$current\_id, {  
 if(input$dataset%in%'Exports'){  
 shiny::req(!is.null(input$current\_id) & stringr::str\_detect(input$current\_id, pattern = "delete"))  
 rv$dt\_row <- which(stringr::str\_detect(rv$df$Buttons, pattern = paste0("\\b", input$current\_id, "\\b")))  
 rv$df <- rv$df[-rv$dt\_row, ]  
 proxy <- DT::dataTableProxy("dt\_table")  
 DT::replaceData(proxy, rv$df, resetPaging = FALSE, rownames = FALSE)  
 }else if(input$dataset%in%'Kenya Debt'){  
 shiny::req(!is.null(input$current\_id) & stringr::str\_detect(input$current\_id, pattern = "delete"))  
 rv1$dt\_row <- which(stringr::str\_detect(rv1$df$Buttons, pattern = paste0("\\b", input$current\_id, "\\b")))  
 rv1$df <- rv1$df[-rv1$dt\_row, ]  
 proxy <- DT::dataTableProxy("dt\_table")  
 DT::replaceData(proxy, rv1$df, resetPaging = FALSE, rownames = FALSE)  
 }else{  
 return()  
 }  
 })  
 shiny::observeEvent(input$current\_id, {  
 if(input$dataset%in%'Exports'){  
 shiny::req(!is.null(input$current\_id) & stringr::str\_detect(input$current\_id, pattern = "edit"))  
 rv$dt\_row <- which(stringr::str\_detect(rv$df$Buttons, pattern = paste0("\\b", input$current\_id, "\\b")))  
 df <- rv$df[rv$dt\_row, ]  
 modal\_dialog(  
 dates= df$dates, type = df$type, crop = df$crop, weight = df$weight, edit = TRUE  
 )  
 rv$add\_or\_edit <- NULL  
 }else if(input$dataset%in%'Kenya Debt'){  
 shiny::req(!is.null(input$current\_id) & stringr::str\_detect(input$current\_id, pattern = "edit"))  
 rv1$dt\_row <- which(stringr::str\_detect(rv1$df$Buttons, pattern = paste0("\\b", input$current\_id, "\\b")))  
 df <- rv1$df[rv1$dt\_row, ]  
 modal\_dialog1(  
 datex= df$date, indicators = df$indicator,value = df$value, edit = TRUE  
 )  
 rv1$add\_or\_edit <- NULL  
 }else{  
 return()  
 }  
 })  
 shiny::observeEvent(input$final\_edit, {  
 req(input$weight)  
 req(input$dates)  
 shiny::req(!is.null(input$current\_id) & stringr::str\_detect(input$current\_id, pattern = "edit") & is.null(rv$add\_or\_edit))  
 rv$edited\_row <- dplyr::tibble(  
 dates= input$dates,  
 type = input$type,  
 crop = input$crop,   
 weight = input$weight,  
 Buttons = rv$df$Buttons[rv$dt\_row]  
 )  
 rv$df[rv$dt\_row, ] <- rv$edited\_row  
   
 })  
 shiny::observeEvent(input$final\_edit,{  
 shinyFeedback::feedbackWarning('weight',is.na(input$weight),"\*Required")  
 req(input$weight)  
 req(input$dates)  
 shiny::req(rv$add\_or\_edit == 1)  
 add\_row <- dplyr::tibble(  
 dates= input$dates,  
 type = input$type,  
 crop = input$crop,   
 weight = input$weight,  
 Buttons = create\_btns(rv$keep\_track\_id)  
 )  
 rv$df <- add\_row %>%  
 dplyr::bind\_rows(rv$df)  
 rv$keep\_track\_id <- rv$keep\_track\_id + 1  
 proxy <- DT::dataTableProxy("dt\_table")  
 shiny::observe({  
 req(input$crop)  
 req(input$dates)  
 req(input$weight)  
 DT::replaceData(proxy, rv$df, resetPaging = FALSE, rownames = FALSE)  
 })  
 })  
 shiny::observeEvent(input$dismiss\_modal, {  
 shiny::removeModal()  
 })  
 shiny::observeEvent(input$final\_edit, {  
 req(input$crop)  
 req(input$weight)  
 req(input$dates)  
 shiny::removeModal()  
 })  
   
 shiny::observeEvent(input$final\_edit1, {  
 req(input$value)  
 req(input$datex)  
 req(input$indicators)  
 shiny::req(!is.null(input$current\_id) & stringr::str\_detect(input$current\_id, pattern = "edit") & is.null(rv1$add\_or\_edit))  
 rv1$edited\_row <- dplyr::tibble(  
 date= input$datex,  
 indicator = input$indicators,  
 value = input$value,  
 Buttons = rv1$df$Buttons[rv1$dt\_row]  
 )  
 rv1$df[rv1$dt\_row, ] <- rv1$edited\_row  
 })  
 shiny::observeEvent(input$final\_edit1,{  
 shinyFeedback::feedbackWarning('value',is.na(input$value),"\*Required")  
 req(input$value)  
 req(input$datex)  
 shiny::req(rv1$add\_or\_edit == 1)  
 add\_row <- dplyr::tibble(  
 date= input$datex,  
 indicator = input$indicators,  
 value = input$value,  
 Buttons = create\_btns(rv1$keep\_track\_id)  
 )  
 rv1$df <- add\_row %>%  
 dplyr::bind\_rows(rv1$df)  
 rv1$keep\_track\_id <- rv1$keep\_track\_id + 1  
 proxy <- DT::dataTableProxy("dt\_table")  
 shiny::observe({  
 req(input$value)  
 req(input$datex)  
 req(input$indicators)  
 DT::replaceData(proxy, rv1$df, resetPaging = FALSE, rownames = FALSE)  
 })  
 })  
 shiny::observeEvent(input$dismiss\_modal1, {  
 shiny::removeModal()  
 })  
 shiny::observeEvent(input$final\_edit1, {  
 req(input$value)  
 req(input$datex)  
 req(input$indicators)  
 shiny::removeModal()  
 })  
   
 shiny::observeEvent(input$update, {  
 if(input$dataset%in%'Exports'){  
 showModal(modalDialog(  
 tagList('Updating overwrites the existing Exports Data'),  
 title='Update Exports Data',  
 footer=tagList(loadingButton('confirm','Update Data',   
 class = "btn-info",  
 loadingLabel = "Updating..."),  
 modalButton('Cancel')),  
 size = 'm',  
 easyClose = TRUE  
 )  
 )  
 }else if(input$dataset%in%'Kenya Debt'){  
 showModal(modalDialog(  
 tagList('Updating overwrites the existing Kenya Debt Records'),  
 title='Update Kenya Debt Records',  
 footer=tagList(loadingButton('confirm1','Update Data',   
 class = "btn-danger",  
 loadingLabel = "Updating..."),  
 modalButton('Cancel')),  
   
 size = 'm',  
 easyClose = TRUE  
 )  
 )  
 }else{  
 return()  
 }  
 })  
   
 observeEvent(input$confirm,{  
 final<-subset(rv$df, select = -c(Buttons))  
 openxlsx::write.xlsx(final,  
 file ='~/Programming/R/DATA/exports1.xlsx',  
 colNames = TRUE, borders = "columns")  
 resetLoadingButton("confirm")  
 removeModal()  
 showToast(  
 "success",   
 "EXPORTS DATA UPDATED",   
 .options = myToastOptions  
 )  
 })  
 observeEvent(input$confirm1,{  
 final1<-subset(rv1$df, select = -c(Buttons))  
 openxlsx::write.xlsx(final1,  
 file = '~/Programming/R/DATA/kenya\_status1.xlsx',  
 colNames = TRUE, borders = "columns")  
 resetLoadingButton("confirm1")  
 removeModal()  
 showToast(  
 "success",   
 "DEBT RECORDS UPDATED",   
 .options = myToastOptions  
 )  
 removeModal()  
 })  
   
 verify<- modalDialog(  
 tags$h2('ARE YOU A DATABASE ADMINISTRATOR AT INFINICALS?'),  
 title=tags$h3('DATA ADMINISTATOR'),  
 footer=tagList(actionButton('accept','YES', class = "btn-info"),  
 actionButton('decline','NO', class = "btn-info")),  
 size = 'm',  
 easyClose = FALSE)  
   
 observeEvent(input$tabs, {  
 showModal(verify)  
 }, once=TRUE,ignoreInit=TRUE  
 )  
   
 shiny::hideTab('tabs',target = 'Authenticate',session=session)  
   
 observeEvent(input$decline,{  
 shiny::hideTab('tabs',target = 'Data',session=session)  
 shiny::hideTab('tabs',target = 'View Available Data',session=session)  
 shiny::showTab('tabs',target = 'Authenticate',session=session)  
 shiny::removeModal()  
 })  
 observeEvent(input$accept,{  
 shinyalert::shinyalert(title='INFINICALS PASSCODE',  
 html = TRUE,  
 showConfirmButton = FALSE,  
 animation = "pop",  
 imageUrl= 'logo.png',  
 text = tagList(  
 passwordInput('passcode','Passcode:',  
 placeholder='Enter official passcode'),  
 actionButton('ok','OK')  
 ))  
 })  
 observeEvent(input$ok,{  
 if(input$passcode==2140){  
 shinyalert::closeAlert()  
 shiny::removeModal()  
 shiny::showTab('tabs',target = 'Data',session=session)  
 shiny::hideTab('tabs',target = 'View Available Data',session=session)  
 shiny::hideTab('tabs',target = 'Authenticate',session=session)  
 } else{  
 shinyalert::closeAlert()  
 shiny::removeModal()  
 shiny::hideTab('tabs',target = 'Data',session=session)  
 shiny::hideTab('tabs',target = 'View Available Data',session=session)  
 shiny::showTab('tabs',target = 'Authenticate',session=session)  
 output$base<- renderText({  
 validate(need(input$passcode==2140, paste(input$userName, "is not Verified!. You are not allowed to access the Company Database.",sep=' ')))  
 })  
 }  
 })  
   
 observeEvent(input$go,{  
 showModal(verify)  
 })  
 observeEvent(input$add\_export, {  
 if(input$dataset %in%'Kenya Debt'){  
 req(input$datex)  
 req(input$indicators)  
 observeEvent(input$datex, {  
 req(input$datex)  
 req(input$indicators)  
 day <-rv1$df %>% dplyr::filter(grepl(input$datex,date))%>%  
 filter(stringr::str\_detect(indicator,'domestic\_debt'))  
 day1 <- rv1$df %>% dplyr::filter(grepl(input$datex,date))%>%  
 filter(stringr::str\_detect(indicator,'external\_debt'))  
 day2<- rv1$df %>% dplyr::filter(grepl(input$datex,date))%>%  
 filter(stringr::str\_detect(indicator,'total'))  
 if(nrow(day)==0 & nrow(day1)==0 & nrow(day2)==0){  
 enableActionButton("final\_edit1",session)  
 updateSelectInput(inputId = "indicators",  
 choices =c('domestic\_debt','external\_debt'))  
 }else if(nrow(day)==1 & nrow(day1)==0 & nrow(day2)==0) {  
 enableActionButton("final\_edit1",session)  
 updateSelectInput(inputId = "indicators",  
 choices ='external\_debt')  
 }else if(nrow(day)==0 & nrow(day1)==1 & nrow(day2)==0) {  
 enableActionButton("final\_edit1",session)  
 updateSelectInput(inputId = "indicators",  
 choices ='domestic\_debt')  
 }else if(nrow(day)==1 & nrow(day1)==1 & nrow(day2)==0){  
 enableActionButton("final\_edit1",session)  
 subs<- rv1$df$date  
 subs1<-rv1$df%>%dplyr::filter(grepl(input$datex,date))%>%  
 select(indicator,value)%>%filter(indicator%in%'domestic\_debt')%>%select(value)  
 s<- subs1[[1,1]]  
 subs2<-rv1$df%>%dplyr::filter(grepl(input$datex,date))%>%  
 select(indicator,value)%>%filter(indicator%in%'external\_debt')%>%select(value)  
 t<-subs2[[1]]  
 total=s+t  
 updateSelectInput(inputId = "indicators",  
 choices ='total')  
 updateNumericInput(session = session,inputId='value',  
 value=total)  
 }else if(nrow(day)==1 & nrow(day1)==1 & nrow(day2)==1){  
 updateSelectInput(inputId = "indicators",  
 choices ='')  
 disableActionButton("final\_edit1",session)  
 }else{  
 return()  
 }  
 })  
 }  
 })  
   
 }  
   
 shinyApp(ui = ui, server = server)