library(shiny)

library(imager)

library(shinydashboard)

library(shinyMatrix)

library(shinyjs)

library(shinyEventLogger)

library(base64enc)

library(geometry)

library(image.darknet)

library(ECharts2Shiny)

set\_logging()

log\_init()

box1 <<- c(0,0,0,0)

box2 <<- c(0,0,0,0)

ui <- dashboardPage(

dashboardHeader(title = "Exam Center"),

dashboardSidebar(sidebarMenu(

menuItem("Home", tabName = "Home", icon = icon("table")),

menuItem("Settings", tabName = "readMe", icon = icon("cog"))

)),

dashboardBody(

useShinyjs(),

loadEChartsLibrary(),

tabItems(

tabItem(tabName = "Home",

fluidRow(column(

8,

offset = 2,

box(

id = "myVideo",

width = 12,

height = NULL ,

collapsible = T,

solidHeader = T,

tags$head(

tags$link(rel = "stylesheet", type = "text/css", href = "style.css")

),

div(

align = 'center',

tags$video(

id = "video",

autoplay = T,

width = 640,

height = 480

),

# hidden(tags$canvas(

# id = "canvas", width = 640, height = 480

# )),

tags$canvas(

id = "canvas", width = 640, height = 480

),

hidden(textInput('imgStr', 'Label')),

p(),

actionButton('startExam', 'Start'),

includeScript(path = "video.js"),

sliderInput(

"threshold",

"Threshold",

min = 0,

max = 100,

value = 98,

step = 1

),

)

),

),),

fluidRow(column(

8,

offset = 2,

valueBoxOutput("result", width = 12),

),),),

tabItem(tabName = "Settings",)

),

)

)

server <- function(input, output, session) {

set\_logging\_session()

useShinyjs()

options(shiny.maxRequestSize = 30 \* 1024 ^ 2)

getDiff <- function(image1, image2) {

img1 <- load.image(image1)[, , , 1]

img2 <- load.image(image2)[, , , 1]

img1[box1[1]:box1[2], box1[3]:box1[4]] = 0

img2[box2[1]:box2[2], box2[3]:box2[4]] = 0

img1cropped <- data.matrix(img1 [img1 != 0 & img2 != 0])

img2cropped <- data.matrix(img2[img1 != 0 & img2 != 0])

dotProd <- dot(img1cropped, img2cropped)

magImg1 <- norm(img1cropped, type = "F")

magImg2 <- norm(img2cropped, type = "F")

cosSim <- dotProd / (magImg1 \* magImg2)

return (cosSim)

return(1)

}

getBox <- function(image) {

yolo\_tiny\_voc <-

image\_darknet\_model(

type = "detect",

model = "tiny-yolo-voc.cfg",

weights = system.file(package = "image.darknet", "models", "tiny-yolo-voc.weights"),

labels = system.file(package = "image.darknet", "include", "darknet", "data",

"voc.names")

)

x <-

image\_darknet\_detect(

file = paste(getwd(), image, sep = "/"),

object = yolo\_tiny\_voc,

threshold = 0.19

)

detected <- load.image("predictions.png")

blue\_pred <- detected[, , , 3]

top <- -1

bot <- -1

left <- -1

right <- -1

for (i in 1:dim(blue\_pred)[1]) {

if (any(0 == detected[, , , 1][i,] & (127/255) == detected[, , , 2][i,] & 1 == detected[, , , 3][i,]) & left == -1) {

left <- i - 1

} else if (all(0 != detected[, , , 1][i,] & (127/255) != detected[, , , 2][i,] & 1 != detected[, , , 3][i,]) &

left != -1 & right == -1) {

right <- i

print(i)

}

}

if (left != -1 & right == -1) {

right <- dim(blue\_pred)[1]

}

for (i in 1:dim(blue\_pred)[2]) {

if (any(0 == detected[, , , 1][,i] & (127/255) == detected[, , , 2][,i] & 1 == detected[, , , 3][,i]) & top == -1) {

top <- i - 1

} else if (all(0 != detected[, , , 1][,i] & (127/255) != detected[, , , 2][,i] & 1 != detected[, , , 3][,i]) & top != -1 & bot == -1) {

bot <- i

}

}

if (top != -1 & bot == -1) {

bot <- dim(blue\_pred)[2]

}

return (c(left, right, top, bot))

}

img\_str <- reactive({

input$imgStr

})

observeEvent(input$startExam, {

first <<- TRUE

cosineSimilarity <<- 1

observe({

invalidateLater(60000, session)

isolate({

runjs(

'

var video = document.getElementById("video");

console.log("hi")

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

context.drawImage(video, 0, 0);

str2 = canvas.toDataURL().split(",").pop();

//document.getElementById("imgStr").value=str2;

console.log(str2);

Shiny.setInputValue("imgStr", str2);

'

)

img <- input$imgStr

if (first == FALSE) {

outconn <- file("img\_current.jpg", "wb")

base64decode(what = img, output = outconn)

close(outconn)

Sys.sleep(1)

box2<<-getBox("img\_current.jpg")

print("current img created")

print("box2")

print(box2)

}

if (first == TRUE & img != "") {

outconn <- file("img\_first.jpg", "wb")

base64decode(what = img, output = outconn)

close(outconn)

first <<- FALSE

Sys.sleep(1)

box1<<- getBox("img\_first.jpg")

print("first img created")

print("box1")

print(box1)

}

if (file.exists("img\_first.jpg") & file.exists("img\_current.jpg")) {

cosineSimilarity <<- getDiff("img\_first.jpg", "img\_current.jpg")

log\_event(cosineSimilarity)

}

if (cosineSimilarity >= input$threshold / 100) {

output$result <- renderValueBox({

valueBox(

paste0(round(cosineSimilarity \* 100, 1), "%"),

"Cosine Similarity",

icon = icon("fas fa-thumbs-up"),

color = "green"

)

})

} else {

output$result <- renderValueBox({

valueBox(

paste0(round(cosineSimilarity \* 100, 1), "%"),

"Cosine Similarity",

icon = icon("fas fa-thumbs-down"),

color = "red"

)

})

}

})

})

})

}

shinyApp(ui, server)