**# Initialize renv for dependency management**

**# install.packages("renv")**

**# renv::init()**

**# install.packages(c("shiny", "dplyr", "DT", "stringr", "shinyjs", "httr", "jsonlite", "markdown"))**

**# renv::snapshot()**

**library(shiny)**

**library(dplyr)**

**library(DT)**

**library(stringr)**

**library(shinyjs)**

**library(httr)**

**library(jsonlite)**

**library(markdown)**

**# Suppress package warnings**

**suppressPackageStartupMessages({**

**library(dplyr)**

**library(DT)**

**})**

**# GitHub raw URL**

**github\_base\_url <- "https://raw.githubusercontent.com/khandangit/ISPMExplorer/main/ISPM\_files/"**

**# ISPM metadata (80 PDFs: 40 ISPMs, 40 guidelines)**

**ispm\_data <- data.frame(**

**ISPM\_Number = c(1:40, rep(NA, 40)), # ISPMs 1–40, NA for guidelines**

**Link = c(**

**paste0("ISPM\_", 1:40, ".pdf"), # ISPMs**

**paste0("GUIDELINE\_ISPM\_", 1:40, ".pdf") # Guidelines**

**),**

**Title = c(**

**# ISPMs 1–20 (existing)**

**"Phytosanitary principles for plant protection",**

**"Framework for pest risk analysis",**

**"Code for biological control agents",**

**"Pest free areas requirements",**

**"Glossary of phytosanitary terms",**

**"Guidelines for surveillance",**

**"Export certification system",**

**"Pest status determination",**

**"Pest eradication guidelines",**

**"Pest free production sites",**

**"Quarantine pest risk analysis",**

**"Biological control agents guidelines",**

**"Phytosanitary import regulations",**

**"Inspection guidelines",**

**"Wood packaging regulation",**

**"Integrated measures for plants",**

**"Commodity pest risk categorization",**

**"Equivalence of phytosanitary measures",**

**"Diagnostic protocols for pests",**

**"Regulated pest lists guidelines",**

**# ISPMs 21–40 (placeholders)**

**paste0("ISPM ", 21:40, " Standard"),**

**# Guidelines 1–40 (placeholders)**

**paste0("Guideline for ISPM ", 1:40)**

**),**

**Category = c(**

**rep("Pest Risk Analysis", 10),**

**rep("Surveillance", 10),**

**rep("Quarantine", 10),**

**rep("Certification", 10), # ISPMs**

**rep("Guidelines", 40) # Guidelines**

**),**

**stringsAsFactors = FALSE**

**)**

**# Add full PDF URLs**

**ispm\_data$PDF\_URL <- paste0(github\_base\_url, ispm\_data$Link)**

**# Verify ispm\_data**

**cat("ispm\_data rows:", nrow(ispm\_data), "\n")**

**cat("ispm\_data columns:", paste(colnames(ispm\_data), collapse = ", "), "\n")**

**# Function to hyperlink ISPM references**

**hyperlink\_references <- function(text, data) {**

**# Log original text**

**cat("Original response text:", text, "\n")**

**# Hyperlink ISPM X (handle variations: ISPM X, ISPM-X, ISPM No. X)**

**for (i in 1:40) {**

**pattern <- paste0("\\b(ISPM\\s\*(?:No\\.?\\s\*)?", i, "|ISPM\\s\*-\\s\*", i, ")\\b")**

**replacement <- sprintf("[%s](%s)", paste("ISPM", i), data$PDF\_URL[data$ISPM\_Number == i])**

**matches <- gregexpr(pattern, text, ignore.case = TRUE)[[1]]**

**if (matches[1] != -1) {**

**cat("ISPM", i, "matches found at positions:", matches, "\n")**

**}**

**text <- gsub(pattern, replacement, text, ignore.case = TRUE)**

**}**

**# Hyperlink Guideline ISPM X**

**for (i in 1:40) {**

**pattern <- paste0("\\b(Guideline\\s\*ISPM\\s\*(?:No\\.?\\s\*)?", i, "|Guideline\\s\*ISPM\\s\*-\\s\*", i, ")\\b")**

**replacement <- sprintf("[%s](%s)", paste("Guideline ISPM", i), data$PDF\_URL[is.na(data$ISPM\_Number) & grepl(paste0("GUIDELINE\_ISPM\_", i), data$Link)])**

**matches <- gregexpr(pattern, text, ignore.case = TRUE)[[1]]**

**if (matches[1] != -1) {**

**cat("Guideline ISPM", i, "matches found at positions:", matches, "\n")**

**}**

**text <- gsub(pattern, replacement, text, ignore.case = TRUE)**

**}**

**cat("Hyperlinked response text:", text, "\n")**

**return(text)**

**}**

**# Function to call OpenAI API**

**query\_openai <- function(question, api\_key, context = "") {**

**cat("Calling OpenAI API for question:", question, "\n")**

**if (api\_key == "") {**

**cat("API key missing\n")**

**return(list(success = FALSE, response = "Error: OpenAI API key not set. Please configure OPENAI\_API\_KEY in .Renviron."))**

**}**

**tryCatch({**

**response <- POST(**

**url = "https://api.openai.com/v1/chat/completions",**

**add\_headers(Authorization = paste("Bearer", api\_key)),**

**content\_type\_json(),**

**encode = "json",**

**body = list(**

**model = "gpt-4.1-nano",**

**messages = list(**

**list(role = "system", content = paste(**

**"You are an expert on International Standards for Phytosanitary Measures (ISPMs). Provide detailed answers (200–300 words) with specific ISPM references and examples. Format ALL references to ISPMs and guidelines as Markdown hyperlinks, e.g., [ISPM 20](https://...) or [Guideline ISPM 20](https://...), including in-text mentions like 'ISPM 20' or 'Guideline ISPM 20'. Use only the provided context or knowledge of ISPMs 1-40 and their guidelines. Context:", context**

**)),**

**list(role = "user", content = question)**

**),**

**max\_tokens = 500,**

**temperature = 0.7**

**)**

**)**

**cat("OpenAI API status:", status\_code(response), "\n")**

**if (status\_code(response) == 200) {**

**content <- content(response, as = "parsed")**

**answer <- content$choices[[1]]$message$content**

**# Hyperlink plain references**

**answer <- hyperlink\_references(answer, ispm\_data)**

**cat("OpenAI response content:", answer, "\n")**

**return(list(success = TRUE, response = answer))**

**} else {**

**error\_msg <- content(response)$error$message %||% "Unknown error"**

**cat("OpenAI API error:", error\_msg, "\n")**

**if (status\_code(response) == 429) {**

**error\_msg <- paste(error\_msg, "Please check your OpenAI plan at https://platform.openai.com/account/billing.")**

**}**

**return(list(success = FALSE, response = paste("API error:", status\_code(response), error\_msg)))**

**}**

**}, error = function(e) {**

**cat("OpenAI API call failed:", e$message, "\n")**

**return(list(success = FALSE, response = paste("Error calling OpenAI API:", e$message)))**

**})**

**}**

**# Format ISPM context for OpenAI with query-based filtering**

**format\_ispm\_context <- function(data, query = NULL) {**

**if (!is.null(query)) {**

**data <- data %>%**

**filter(grepl(tolower(query), tolower(Title)) | grepl(tolower(query), tolower(Category)))**

**cat("Filtered context rows:", nrow(data), "\n")**

**}**

**paste(**

**sapply(1:nrow(data), function(i) {**

**num <- ifelse(is.na(data$ISPM\_Number[i]), "Guideline", paste("ISPM", data$ISPM\_Number[i]))**

**paste0(**

**num, ": ", str\_trunc(data$Title[i], 20),**

**" (Category: ", data$Category[i], ", PDF: ", data$PDF\_URL[i], ")"**

**)**

**}),**

**collapse = "\n"**

**)**

**}**

**# Truncate title for dropdown**

**format\_dropdown\_title <- function(ispm\_num, title) {**

**label <- ifelse(is.na(ispm\_num), "Guideline", paste("ISPM", ispm\_num))**

**words <- unlist(strsplit(title, "\\s+"))**

**truncated <- if (length(words) > 10) {**

**paste(head(words, 10), collapse = " ")**

**} else {**

**title**

**}**

**paste0(label, ": ", truncated)**

**}**

**# UI**

**ui <- fluidPage(**

**useShinyjs(),**

**tags$head(**

**tags$style(HTML("**

**body { background-color: #f4f6f9; font-family: Arial, sans-serif; }**

**.title-panel { background-color: #005a87; color: white; padding: 10px; border-radius: 5px; text-align: center; }**

**.sidebar { background-color: #e6f0fa; padding: 15px; border-radius: 5px; }**

**.btn-primary { background-color: #28a745; border-color: #28a745; border-radius: 5px; }**

**.btn-primary:hover { background-color: #218838; border-color: #218838; }**

**.dataTable th { background-color: #005a87; color: white; }**

**.dataTable tr:nth-child(even) { background-color: #f8f9fa; }**

**.dataTable tr:hover { background-color: #e9ecef; }**

**.disclaimer { border: 1px solid #ccc; padding: 10px; border-radius: 5px; background-color: #fff; font-size: 12px; }**

**a { color: #005a87; }**

**a:hover { color: #003d5b; }**

**.dataTables\_wrapper { max-width: 1200px; margin: 0 auto; width: 100% !important; }**

**.dataTable { width: 100% !important; }**

**.qa-container { max-height: 400px; overflow-y: auto; border: 1px solid #ccc; padding: 10px; border-radius: 5px; background-color: #fff; }**

**.qa-message { margin-bottom: 20px; }**

**.user-message { color: #005a87; }**

**.answer-message { color: #333; }**

**.footer { text-align: center; padding: 10px; font-size: 12px; color: #666; }**

**")),**

**tags$script(HTML("**

**$(document).ready(function() {**

**$('#qa\_input').keypress(function(e) {**

**if (e.which == 13) { // Enter key**

**$('#qa\_submit').click();**

**e.preventDefault();**

**}**

**});**

**});**

**"))**

**),**

**div(class = "title-panel", titlePanel("ISPM Explorer")),**

**tabsetPanel(**

**tabPanel("Q&A",**

**fluidRow(**

**column(12,**

**textOutput("qa\_status"),**

**br(),**

**div(class = "qa-container",**

**uiOutput("qa\_output")),**

**textInput("qa\_input", "Ask a question:", ""),**

**actionButton("qa\_submit", "Send", class = "btn-primary")**

**)**

**)**

**),**

**tabPanel("Search",**

**sidebarLayout(**

**sidebarPanel(**

**class = "sidebar",**

**p("ISPM Explorer lets you search and filter International Standards for Phytosanitary Measures and guidelines by keyword, number, or category, with downloadable results."),**

**textInput("search", "Search ISPMs/Guidelines (by keyword in title/category):", ""),**

**selectizeInput("ispm\_num", "Filter by ISPM Number or Guideline:",**

**choices = c("All" = "All", setNames(**

**seq\_len(nrow(ispm\_data)),**

**mapply(format\_dropdown\_title, ispm\_data$ISPM\_Number, ispm\_data$Title)**

**)), options = list(maxOptions = 100)),**

**selectInput("category", "Filter by Category:", choices = c("All", unique(ispm\_data$Category))),**

**downloadButton("download", "Download Results", class = "btn-primary"),**

**br(),**

**actionButton("show\_disclaimer", "View Disclaimer", class = "btn-primary"),**

**conditionalPanel(**

**condition = "input.show\_disclaimer % 2 == 1",**

**div(**

**class = "disclaimer",**

**h4("FAO/IPPC Terms of Use"),**

**p("The designations employed and the presentation of material in this information product do not imply the expression of any opinion whatsoever on the part of the Food and Agriculture Organization of the United Nations (FAO) concerning the legal or development status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries. The mention of specific companies or products of manufacturers, whether or not these have been patented, does not imply that these have been endorsed or recommended by FAO in preference to others of a similar nature that are not mentioned."),**

**p("The views expressed in this information product are those of the author(s) and do not necessarily reflect the views or policies of FAO."),**

**p("© FAO, 2024"),**

**p("This work is made available under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 IGO licence (CC BY-NC-SA 3.0 IGO; ", a(href = "https://creativecommons.org/licenses/by-nc-sa/3.0/igo/legalcode", "https://creativecommons.org/licenses/by-nc-sa/3.0/igo/legalcode", target = "\_blank"), ")."),**

**p("Under the terms of this licence, this work may be copied, redistributed and adapted for non-commercial purposes, provided that the work is appropriately cited. In any use of this work, there should be no suggestion that FAO endorses any specific organization, products or services. The use of the FAO logo is not permitted. If the work is adapted, then it must be licensed under the same or equivalent Creative Commons licence. If a translation of this work is created, it must include the following disclaimer along with the required citation: 'This translation was not created by the Food and Agriculture Organization of the United Nations (FAO). FAO is not responsible for the content or accuracy of this translation. The original English edition shall be the authoritative edition.'"),**

**p("Disputes arising under the licence that cannot be settled amicably will be resolved by mediation and arbitration as described in Article 8 of the licence except as otherwise provided herein. The applicable mediation rules will be the mediation rules of the World Intellectual Property Organization ", a(href = "https://www.wipo.int/amc/en/mediation/rules", "https://www.wipo.int/amc/en/mediation/rules", target = "\_blank"), " and any arbitration will be conducted in accordance with the Arbitration Rules of the United Nations Commission on International Trade Law (UNCITRAL)."),**

**p("Third-party materials: Users wishing to reuse material from this work that is attributed to a third party, such as tables, figures or images, are responsible for determining whether permission is needed for that reuse and for obtaining permission from the copyright holder. The risk of claims resulting from infringement of any third-party-owned component in the work rests solely with the user."),**

**p("Sales, rights and licensing: FAO information products are available on the FAO website (", a(href = "https://www.fao.org/publications", "www.fao.org/publications", target = "\_blank"), ") and can be purchased through ", a(href = "mailto:publications-sales@fao.org", "publications-sales@fao.org"), ". Requests for commercial use should be submitted via: ", a(href = "https://www.fao.org/contact-us/licence-request", "www.fao.org/contact-us/licence-request", target = "\_blank"), ". Queries regarding rights and licensing should be submitted to: ", a(href = "mailto:copyright@fao.org", "copyright@fao.org"), ".")**

**)**

**)**

**),**

**mainPanel(**

**DTOutput("ispm\_table")**

**)**

**)**

**)**

**),**

**div(class = "footer",**

**HTML("Developed by HANK, contact: <a href='mailto:khandangithub@gmail.com'>khandangithub@gmail.com</a>")**

**)**

**)**

**# Server**

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

**# Reactive values**

**qa\_history <- reactiveVal(list())**

**qa\_cache <- reactiveVal(list())**

**# OpenAI API key**

**api\_key <- Sys.getenv("OPENAI\_API\_KEY")**

**cat("API key loaded:", ifelse(api\_key == "", "FALSE", "TRUE"), "\n")**

**# ISPM context**

**ispm\_context <- reactive({**

**format\_ispm\_context(ispm\_data, input$qa\_input)**

**})**

**# Update Q&A display**

**update\_qa <- function(user\_msg = NULL, bot\_msg = NULL) {**

**history <- isolate(qa\_history())**

**if (!is.null(user\_msg)) {**

**history <- append(history, list(list(type = "user", msg = user\_msg)))**

**}**

**if (!is.null(bot\_msg)) {**

**history <- append(history, list(list(type = "bot", msg = bot\_msg)))**

**}**

**if (length(history) > 100) {**

**history <- history[(length(history) - 99):length(history)]**

**}**

**qa\_history(history)**

**cat("Updating Q&A history, length:", length(history), "\n")**

**output$qa\_output <- renderUI({**

**HTML(paste(sapply(history, function(h) {**

**if (h$type == "user") {**

**paste("<div class='qa-message user-message'><strong>Question:</strong>", h$msg, "</div>")**

**} else {**

**# Render Markdown to HTML**

**paste("<div class='qa-message answer-message'><strong>Answer:</strong><br>", markdownToHTML(text = h$msg, fragment.only = TRUE), "</div>")**

**}**

**}), collapse = ""))**

**})**

**}**

**# Handle Q&A submission**

**observeEvent(input$qa\_submit, {**

**req(input$qa\_input)**

**query <- input$qa\_input**

**cat("Q&A submit triggered for query:", query, "\n")**

**# Check cache**

**cache <- isolate(qa\_cache())**

**if (!is.null(cache[[query]])) {**

**cat("Cache hit for query:", query, "\n")**

**update\_qa(user\_msg = query, bot\_msg = cache[[query]])**

**updateTextInput(session, "qa\_input", value = "")**

**output$qa\_status <- renderText("Query successful (cached).")**

**return()**

**}**

**result <- query\_openai(query, api\_key, ispm\_context())**

**if (result$success) {**

**cache[[query]] <- result$response**

**qa\_cache(cache)**

**}**

**update\_qa(user\_msg = query, bot\_msg = result$response)**

**updateTextInput(session, "qa\_input", value = "")**

**output$qa\_status <- renderText(ifelse(result$success, "Query successful.", paste("Query failed:", result$response)))**

**# Force UI refresh**

**shiny::invalidateLater(100, session)**

**}, ignoreInit = TRUE)**

**# Reactive for search and filter**

**filtered\_data <- reactive({**

**cat("Computing filtered\_data, input search:", input$search %||% "", ", ispm\_num:", input$ispm\_num %||% "All", ", category:", input$category %||% "All", "\n")**

**data <- ispm\_data**

**if (!is.null(input$ispm\_num) && input$ispm\_num != "All" && !is.na(input$ispm\_num)) {**

**row\_idx <- as.numeric(input$ispm\_num)**

**data <- data[row\_idx, , drop = FALSE]**

**cat("Filtered by ISPM/Guideline, rows:", nrow(data), "\n")**

**}**

**if (!is.null(input$category) && input$category != "All" && !is.na(input$category)) {**

**data <- data %>% filter(Category == input$category)**

**cat("Filtered by Category, rows:", nrow(data), "\n")**

**}**

**if (!is.null(input$search) && input$search != "" && !is.na(input$search)) {**

**data <- data %>%**

**filter(**

**grepl(tolower(input$search), tolower(Title), fixed = TRUE) |**

**grepl(tolower(input$search), tolower(Category), fixed = TRUE)**

**)**

**cat("Filtered by search keyword, rows:", nrow(data), "\n")**

**}**

**cat("Final filtered data rows:", nrow(data), "\n")**

**if (nrow(data) == 0) {**

**data <- data.frame(ISPM\_Number = integer(), Title = character(), Category = character(), Link = character())**

**}**

**data %>% select(ISPM\_Number, Title, Category, Link)**

**})**

**# Debug filtered\_data**

**observe({**

**cat("Filtered data updated, rows:", nrow(filtered\_data()), "\n")**

**})**

**# Render table**

**output$ispm\_table <- renderDT({**

**data <- filtered\_data()**

**cat("Rendering DT table with rows:", data, "\n")**

**if (nrow(data) == 0) {**

**return(datatable(data.frame(Message = "No results found"), options = list(dom = 't')))**

**}**

**data$Link <- sprintf('<a href="%s%s" target="\_blank">View PDF</a>', github\_base\_url, data$Link)**

**datatable(**

**data,**

**escape = FALSE,**

**options = list(**

**pageLength = 10,**

**dom = 'tip'**

**)**

**)**

**})**

**# Download handler**

**output$download <- downloadHandler(**

**filename = function() {**

**paste("ispm\_search\_results\_", Sys.Date(), ".csv", sep = "")**

**},**

**content = function(file) {**

**data <- filtered\_data()**

**if (nrow(data) == 0) {**

**data <- data.frame(Message = "No results found")**

**} else {**

**data$Link <- paste0(github\_base\_url, data$Link)**

**}**

**write.csv(data, file, row.names = FALSE, fileEncoding = "UTF-8")**

**}**

**)**

**}**

**# Run the app**

**shinyApp(ui = ui, server = server)**