# Get all page content for selected team

team\_page <- shiny::reactive({

shiny::req(input$team != " ")

url <- paste0(input$team)

httr2::request(url) |>

httr2::req\_perform() |>

httr2::resp\_body\_html()

})

# Get all page content for selected cheerleader

cheerleader\_page <- shiny::reactive({

shiny::req(input$cheerleader != " ")

url <- paste0(wiki\_url, input$cheerleader)

httr2::request(url) |>

httr2::req\_perform() |>

httr2::resp\_body\_html()

})

cheerleader\_page() is the xml\_nodes object for the entire cheerleader page

we can get hyperlinks, images

page <- httr2::request(url) |>

httr2::req\_perform() |>

httr2::resp\_body\_html()

To get the index of the table, we use rvest::html\_table to return the xml nodes tables as data.frame data. Then search for a value that only appears in the table we want and return the index to subset the xml\_nodes table list.

xml\_tables <- rvest::html\_nodes(page, "table")

df\_tables <- lapply(xml\_tables, rvest::html\_table, fill = TRUE)

keyword <- “unique word found in the table”

for (i in seq\_along(tables)) {

table <- tables[[i]]

if (any(!is.na(table$X1))) {

if (any(table$X1 == keyword)) {

return(i) # return the index of the cheerleader table

}

}

}

bio\_table <- xml\_tables[[index]]

We have the page data for both operations

getCheerleaderPhoto(page)

getCheerleaderBio(page)

We don’t need df\_tables for the cheerleader photo, only xml nodes, but we need df\_tables to get the index of the cheerleader table.

Pages can be reactiveVals() that stores both the xml\_tables and df\_tables object and further refines to the table we actually require for the specified cheerleader biography page.

# Get all page content for selected cheerleader

cheerleader\_page <- shiny::reactive({

shiny::req(input$cheerleader != " ")

url <- paste0(wiki\_url, input$cheerleader)

page <- httr2::request(url) |>

httr2::req\_perform() |>

httr2::resp\_body\_html()

xml\_tables <- rvest::html\_nodes(page, "table")

df\_tables <- lapply(xml\_tables, rvest::html\_table, fill = TRUE)

index <- bioTableIndex(df\_tables, "nationality")

bio\_table <- xml\_tables[[index]]

list(

bio\_table = bio\_table,

df\_tables = df\_tables

)

})

TEAM MEMBER COUNT AVERAGE TENURE SOCIAL MEDIA VIEWS

**YouTube Data API v3**

|  |  |
| --- | --- |
| https://www.googleapis.com/auth/youtube | Manage your YouTube account |
| https://www.googleapis.com/auth/youtube.channel-memberships.creator | See a list of your current active channel members, their current level, and when they became a member |
| https://www.googleapis.com/auth/youtube.force-ssl | See, edit, and permanently delete your YouTube videos, ratings, comments and captions |
| https://www.googleapis.com/auth/youtube.readonly | View your YouTube account |
| https://www.googleapis.com/auth/youtube.upload | Manage your YouTube videos |
| https://www.googleapis.com/auth/youtubepartner | View and manage your assets and associated content on YouTube |
| https://www.googleapis.com/auth/youtubepartner-channel-audit | View private information of your YouTube channel relevant during the audit process with a YouTube partner |

**YouTube Analytics API**

|  |  |
| --- | --- |
| https://www.googleapis.com/auth/youtube | Manage your YouTube account |
| https://www.googleapis.com/auth/youtube.readonly | View your YouTube account |
| https://www.googleapis.com/auth/youtubepartner | View and manage your assets and associated content on YouTube |
| https://www.googleapis.com/auth/yt-analytics-monetary.readonly | View monetary and non-monetary YouTube Analytics reports for your YouTube content |
| https://www.googleapis.com/auth/yt-analytics.readonly | View YouTube Analytics reports for your YouTube content |

**YouTube Reporting API**

|  |  |
| --- | --- |
| https://www.googleapis.com/auth/yt-analytics-monetary.readonly | View monetary and non-monetary YouTube Analytics reports for your YouTube content |
| https://www.googleapis.com/auth/yt-analytics.readonly | View YouTube Analytics reports for your YouTube content |

~~YouTube Ads Reach API~~

~~Freebase API~~

**VOSON SML**

1. Authenticate
2. Collect
3. Create

Verbose = TRUE

WriteToFile = TRUE saves RDS

Voson.data path

Voson.cat TRUE – verbose output printed

Authentication objects created once unless credentials change

Save YouTube authentication objects to file and load them in future sessions

Only does comments

**YTAnalytics**

YTAnalytics::youtube\_oauth(clientId = Sys.getenv("YT\_CLIENT\_ID"),

clientSecret = Sys.getenv("YT\_CLIENT\_SECRET"))

**instaR**

basic/ public content

**tuber::**