```r
library(shiny)
library(plotly)
library(dplyr)
library(ggplot2)
library(DT)
library(Lahman)
library(baseballr)
```

```r
players_data <- Lahman::Batting %>%
  inner_join(Lahman::People, by = "playerID")
savant_data <- read.csv("savant_data.csv", stringsAsFactors = FALSE)
teams_data <- Lahman::Teams
```

```r
ui <- navbarPage("MLB Performance Explorer",
  tabPanel("Player Statistics",
    sidebarLayout(
      sidebarPanel(
        selectInput("player_name", "Select Player", choices = unique(players_data$nameFirst)),
        sliderInput("year_range", "Season Range", min = 1990, max = 2023, value = c(2010, 2023), sep = "
      ),
      mainPanel(
        plotlyOutput("player_plot"),
        DTOutput("player_table")
      ))),
  tabPanel("Heatmaps",
    sidebarLayout(
      sidebarPanel(
        textInput("pitcher_id", "Enter Pitcher MLBAM ID", value = "547973"),
        dateRangeInput("pitch_date", "Pitch Date Range", start = "2023-04-01", end = "2023-10-01"),
        selectInput("pitch_type", "Pitch Type", choices = c("FF", "SL", "CH", "CU", "SI"), selected = "
      ),
      mainPanel(
        plotlyOutput("heatmap_plot"),
        DTOutput("heatmap_data_table")
      )
    )
  ),
  tabPanel("Team Comparison",
    sidebarLayout(
      sidebarPanel(
        sliderInput("team_year", "Select Year", min = 1990, max = 2023, value = 2023, sep = "")
      ),
      mainPanel(
        plotlyOutput("team_barplot"),
        DTOutput("team_table")
      )
    )
  ),
  tabPanel("Historical Explorer",
    sidebarLayout(
      sidebarPanel(
        selectInput("explore_team", "Select Team", choices = unique(teams_data$teamID)),
        selectInput("explore_stat", "Select Stat to Visualize",
                    choices = c("Wins (W)", "Losses (L)", "Attendance"))
```

```
      ),
      mainPanel(
        plotlyOutput("history_plot"),
        DTOutput("history_table")
      )
    )
  )
)
```

```
server <- function(input, output, session) {
  # Player Statistics Plot
  output$player_plot <- renderPlotly({
    filtered <- players_data %>%
      filter(nameFirst == input$player_name, yearID >= input$year_range[1], yearID <= input$year_range[
      group_by(yearID) %>%
      summarise(AVG = mean(H / AB, na.rm = TRUE))

    gg <- ggplot(filtered, aes(x = yearID, y = AVG)) +
      geom_line() + geom_point() +
      labs(title = "Batting Average Over Seasons", x = "Year", y = "AVG")
    ggplotly(gg)
  })
  output$player_table <- renderDT({
    players_data %>%
      filter(nameFirst == input$player_name, yearID >= input$year_range[1], yearID <= input$year_range[
      select(yearID, teamID, AB, H, HR, RBI)
  })
  # Heatmap Placeholder (Statcast)
  output$heatmap_plot <- renderPlotly({
    validate(need(nzchar(input$pitcher_id), "Please enter the pitcher ID"))
    url <- paste0(
      "https://baseballsavant.mlb.com/statcast_search/csv?all=true",
      "&player_type=pitcher",
      "&pitchers_lookup%5B%5D=", input$pitcher_id,
      "&game_date_gt=", input$pitch_date[1],
      "&game_date_lt=", input$pitch_date[2],
      "&type=details"
    )
    pitch_data <- tryCatch({
      read.csv(url, stringsAsFactors = FALSE)
    }, error = function(e) {
      validate(need(FALSE, "Data download failed. Please check the pitcher ID and date range."))
    })
    validate(need(nrow(pitch_data) > 0, "No pitch data found for the selected ID and date range."))
    heat_data <- pitch_data %>%
      filter(pitch_type == input$pitch_type) %>%
      filter(!is.na(plate_x) & !is.na(plate_z))
    validate(need(nrow(heat_data) > 0, "No data available for this pitch type during the selected perio
    gg <- ggplot(heat_data, aes(x = plate_x, y = plate_z)) +
      stat_density2d(aes(fill = ..level..), geom = "polygon", contour = TRUE) +
      scale_fill_viridis_c() +
      labs(
        title = paste("Pitch Location Heatmap -", input$pitch_type),
        x = "Horizontal Location", y = "Vertical Location"
```

```
    )
  ggplotly(gg)
})
output$heatmap_data_table <- renderDT({
  selected_columns <- savant_data %>%
    select(pitch_type, player_name, batter, pitcher)
  datatable(selected_columns)
})
# Team Comparison Plot
output$team_barplot <- renderPlotly({
  team_stats <- teams_data %>%
    filter(yearID == input$team_year) %>%
    mutate(OPS = (H + BB) / AB)
  gg <- ggplot(team_stats, aes(x = reorder(teamID, -OPS), y = OPS)) +
    geom_bar(stat = "identity", fill = "steelblue") +
    labs(title = "Team OPS Comparison", x = "Team", y = "OPS")
  ggplotly(gg)
})
output$team_table <- renderDT({
  teams_data %>%
    filter(yearID == input$team_year) %>%
    select(teamID, W, L, ERA, HR, R)
})

# Historical Explorer Table
output$history_table <- renderDT({
  teams_data %>%
    filter(teamID == input$explore_team) %>%
    select(yearID, divID, W, L, WSWin, attendance) %>%
    arrange(desc(yearID))
})

# Historical Plot
output$history_plot <- renderPlotly({
  filtered_data <- teams_data %>%
    filter(teamID == input$explore_team)

  if (input$explore_stat == "Wins (W)") {
    gg <- ggplot(filtered_data, aes(x = yearID, y = W)) +
      geom_line(color = "blue") +
      geom_point(color = "blue") +
      labs(title = paste(input$explore_team, "Wins Over Seasons"),
           x = "Year", y = "Wins")
  } else if (input$explore_stat == "Losses (L)") {
    gg <- ggplot(filtered_data, aes(x = yearID, y = L)) +
      geom_line(color = "red") +
      geom_point(color = "red") +
      labs(title = paste(input$explore_team, "Losses Over Seasons"),
           x = "Year", y = "Losses")
  } else if (input$explore_stat == "Attendance") {
    gg <- ggplot(filtered_data, aes(x = yearID, y = attendance)) +
      geom_bar(stat = "identity", fill = "steelblue") +
      labs(title = paste(input$explore_team, "Attendance Over Seasons"),
```

```
            x = "Year", y = "Attendance")
    }
    ggplotly(gg)
  })

}
```

```
shinyApp(ui, server)
```