

CampusGives: A Student-Driven Solution to Address Unmet Academic and Financial Needs**Motivation**

Higher education is often seen as a pathway to opportunity, yet for many students, financial struggles create significant barriers to academic success. While tuition and housing costs receive widespread attention, fewer discussions address the everyday academic and material needs of students. Textbooks, study materials, electronics, and dorm essentials are fundamental for learning, yet many students lack the financial means to afford them. Research consistently links financial insecurity to increased stress and declining academic performance (Tindle et al. 2022; Petersen, Louw, and Dumont 2009). Many students resort to loans, part-time jobs, or peer assistance to meet their needs, but these strategies are often inadequate, forcing students to make trade-offs that harm their education.

Beyond financial hardship, the psychological impact of resource scarcity further undermines students' ability to perform well. Studies indicate financial stress weakens cognitive function, reduces motivation, and impairs concentration (Destin and Svoboda 2017). Students facing financial distress often struggle with time management, working memory, and academic engagement, all of which affect their ability to complete coursework effectively (Reid, Jessop, and Miles 2020). Even those receiving financial aid find that grants and stipends do not fully cover essential academic expenses (Huang et al. 2018). This often leads students to take on additional work hours, reducing study time, increasing stress, and perpetuating a cycle that impairs long-term academic success.

Despite these challenges, most university assistance programs focus on financial aid rather than directly addressing students' material needs. Existing solutions rarely provide free, immediate access to essential academic and living supplies, leaving students without the resources required to thrive. CampusGives seeks to bridge this gap by providing a student-driven platform where individuals can list and request items within their campus community. This initiative not only promotes sustainability by redistributing underused resources but also fosters a peer-supported ecosystem that reduces financial barriers in higher education. By enabling students to support one another, CampusGives presents a practical, community-driven solution to a persistent and overlooked problem.

Target Population

The primary beneficiaries of CampusGives are college and university students who face financial challenges in acquiring essential academic and living supplies. This includes students from low-income backgrounds, first-generation college students, and those who rely on financial aid or part-time employment to cover their expenses. The platform specifically aims to support students struggling with textbook costs, access to technology (laptops, calculators), dorm essentials, and other study-related materials.

As a pilot initiative, CampusGives is designed for the University of Texas at Dallas (UTD) student population, where student demand for affordable academic resources is evident. UTD has a diverse student body, including a significant population of international students.

Additionally, CampusGives fosters a peer-to-peer support system, encouraging students with surplus academic materials or dorm supplies to share them with those in need. This not only promotes resource redistribution but also builds a stronger sense of community and mutual support among students. Beyond individual students, the platform can also benefit student organizations, campus sustainability initiatives, and university welfare offices, providing them with a structured system to facilitate the redistribution of resources efficiently.

Methodology

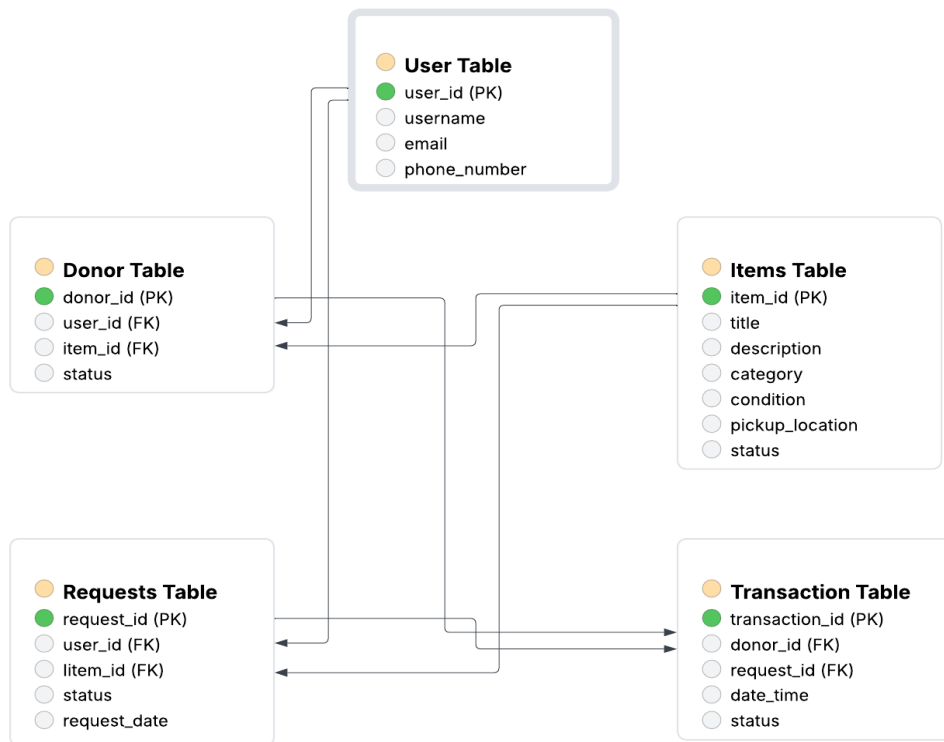
To build CampusGives, I designed and implemented a structured SQL-based database system that helps students exchange academic and personal items easily. The full system was developed and tested using PostgreSQL, Shiny, and SQLite.

Schema Design & Table Creation

I first created a clear schema for the CampusGives platform. The main tables included Users, Items, Donors, Requests, and Transactions. I defined the primary keys, foreign keys, and relationships among them to ensure smooth data flow. These tables were created and structured in pgAdmin, and I manually populated the tables with sample data to support testing.

For example, each donation was stored in the Donor table and linked to a student in the Users table. Each request was recorded in the Requests table, and successful matches were recorded as Transactions. The schema ensures all data is connected clearly and can be tracked from donation to pickup.

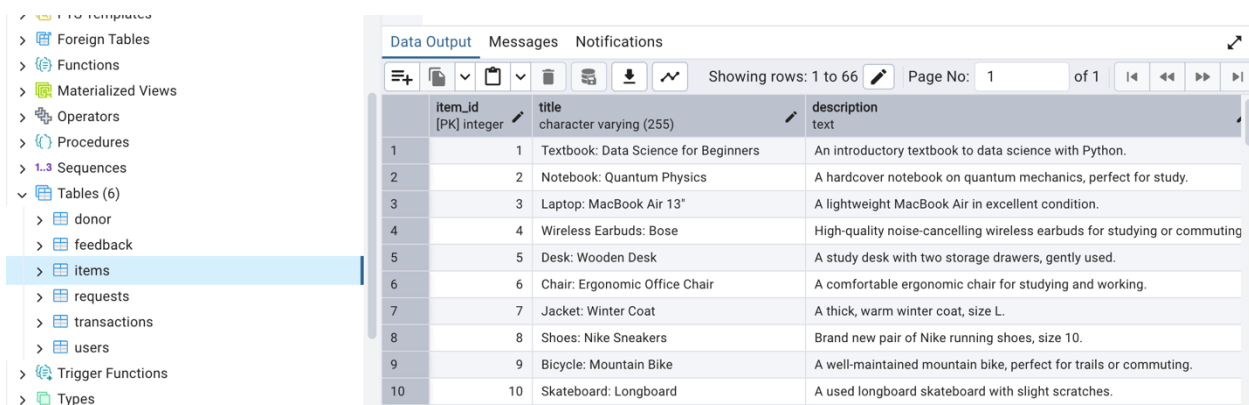
Figure 1: Schema Diagram for CampusGives Data Base



Implementation and Deployment

After I finished building and testing the schema in pgAdmin, I added sample data to each table. This helped me run SQL queries to test if the system could track users, donations, and requests correctly. I used PostgreSQL on localhost to test everything, making sure the database was working well.

After setting up my schema and entering the sample data, I ran multiple test queries and interactions within the Shiny app to make sure everything worked correctly. I tested features like item filtering, making requests, and confirming completed transactions. These tests helped me verify the logic and fix minor issues before publishing.




	item_id [PK] integer	title character varying (255)	description text
1	1	Textbook: Data Science for Beginners	An introductory textbook to data science with Python.
2	2	Notebook: Quantum Physics	A hardcover notebook on quantum mechanics, perfect for study.
3	3	Laptop: MacBook Air 13"	A lightweight MacBook Air in excellent condition.
4	4	Wireless Earbuds: Bose	High-quality noise-cancelling wireless earbuds for studying or commuting.
5	5	Desk: Wooden Desk	A study desk with two storage drawers, gently used.
6	6	Chair: Ergonomic Office Chair	A comfortable ergonomic chair for studying and working.
7	7	Jacket: Winter Coat	A thick, warm winter coat, size L.
8	8	Shoes: Nike Sneakers	Brand new pair of Nike running shoes, size 10.
9	9	Bicycle: Mountain Bike	A well-maintained mountain bike, perfect for trails or commuting.
10	10	Skateboard: Longboard	A used longboard skateboard with slight scratches.

Next, I built the CampusGives interface using R Shiny. The app connects to the database and allows filtering items, request donations, and view transactions. I first tested the Shiny app with the PostgreSQL database. After confirming it worked well, I switched the connection to SQLite so I could publish it easily online.

The Shiny app provides a clean and functional interface for users to interact with the CampusGives donation platform. Users can filter available items by category and condition using dropdown menus, select a date range for transactions, and toggle options like contactless pickup. A row slider lets users control how many entries are shown at a time. The dashboard includes features like data plots and a pickup map to visualize donation activity. After receiving class feedback, I removed some of the interactive tabs and tools to simplify the interface and make it easier for users to navigate.

Finally, I published the full app on shinyapps.io. I also linked the app to my GitHub portfolio site, <https://jpadjadeh.github.io>, so visitors can launch it directly. You can view and interact with it at: <https://jpadjadeh.shinyapps.io/CampusGives/>

 CampusGives Donation App

Filter Items

Choose Category: All

Item Condition: All

[Refresh](#)

Date Range

Filter Transactions: 2025-02-01 to 2025-05-02











☒ Contactless Pickup

Rows to Show:

5 10 50

[Available Items](#) [Requests](#) [Transactions](#) [Pickup Map](#) [Statistics](#)

Show 10 entries Search:

	Photo	title	category	condition	pickup_location
1		Textbook: Data Science for Beginners	Academic & Study Materials	New	Building 1, Room 101
2		Laptop: MacBook Air 13"	Electronic & Computer Accessories	Used	Building 2, Room 202
3		Wireless Earbuds: Bose	Electronic & Computer Accessories	New	Building 2, Room 202
4		Chair: Ergonomic Office Chair	Furniture & Dorm Essentials	New	Building 3, Room 303
5		Jacket: Winter Coat	Clothing & Personal Items	New	Building 4, Room 404
6		Bicycle: Mountain Bike	Transport & Ride-sharing	Used	Building 5, Room 505
7		Skateboard: Longboard	Transport & Ride-sharing	Used	Building 5, Room 505
8		Paint Brushes: Set of 10	Arts & Creative Supplies	New	Building 6, Room 606
9		Blender: Hamilton Beach	Kitchen & Food Supplies	Used	Building 7, Room 707
10		Guitar: Acoustic Guitar	Entertainment & Hobby Items	Used	Building 8, Room 808

Showing 1 to 10 of 40 entries Previous 1 2 3 4 Next

Limitation

One limitation of this project is that I used sample data I created myself for demonstration purposes. While this allowed me to test the system's logic and structure, it may not fully reflect real-world student needs or usage behavior. In future versions, collecting and analyzing real data from students would help improve the accuracy, usefulness, and impact of the system.

Use of AI Tools

I sparingly relied on ChatGPT to help debug and modify sample R Shiny code introduced in class, especially when I encountered challenges adapting it to my project setup. All design decisions, data schema, and implementation were done independently.

Appendix:

R Script

```
# Load libraries
library(shiny)
library(RSQLite)
library(DBI)
library(DT)
library(leaflet)
library(ggplot2)
library(shinyWidgets)
library(shinyjs)
library(shinyalert)
library(stringr)



# Database Connection (SQLite)
con <- dbConnect(RSQLite::SQLite(), dbname = "campusgives.db")






# Fetch categories dynamically
categories <- tryCatch({
  dbGetQuery(con, "
    SELECT DISTINCT i.category
    FROM items i
    JOIN donor d ON i.item_id = d.item_id
    WHERE d.status = 'active'
  ")$category
}, error = function(e) c())
categories <- c("All", categories)

# UI
ui <- fluidPage(
  useShinyjs(),
  useShinyalert(),
  setBackgroundColor("ghostwhite"),
  titlePanel(div(icon("gift", lib = "font-awesome"), "🎁"),
    "CampusGives Donation App")),

  sidebarLayout(
    sidebarPanel(
      h4("🔍 Filter Items"),
      selectInput("category", "Choose Category:", choices =
categories),
```

```

    pickerInput("item_condition", "Item Condition:", choices =
c("New", "Used", "All"), selected = "All", multiple = TRUE, options
= list(`actions-box` = TRUE)),
    actionButton("refresh", " Refresh", class = "btn btn-
primary"),
    hr(),
    h4(" Date Range"),
    dateRangeInput("date_range", "Filter Transactions:", start
= Sys.Date() - 90, end = Sys.Date()),
    checkboxInput("contactless", "Contactless Pickup", TRUE),
    hr(),
    sliderInput("rows_display", "Rows to Show:", min = 5, max =
50, value = 10, step = 5)
  ),

  mainPanel(
    tabsetPanel(
      tabPanel(" Available Items", DTOutput("items_table")),
      tabPanel(" Requests", DTOutput("requests_table")),
      tabPanel(" Transactions",
DTOutput("transactions_table"), br(),
downloadButton("download_receipt", "Download Receipt")),
      tabPanel(" Pickup Map", leafletOutput("pickup_map")),
      tabPanel(" Statistics",
        selectInput("stat_choice", "Choose Statistic:",
choices = c("Transactions Over Time", "Transactions by Status",
"Requests per Day", "Available Items by Category")),
        plotOutput("stats_plot")
      )
    )
  )
)
)
)

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

  get_items <- reactive({
    input$refresh
    isolate({
      query <- "
        SELECT      i.title,          i.category,          i.condition,
i.pickup_location
        FROM items i
        JOIN donor d ON i.item_id = d.item_id
        WHERE d.status = 'active'

```



```

"
  items <- dbGetQuery(con, query)
  if (input$category != "All") items <- items[items$category
== input$category, ]
  if (!"All" %in% input$item_condition) items <-
items[items$condition %in% input$item_condition, ]
  items
})
})

output$items_table <- renderDT({
  items <- get_items()
  placeholder <-
"https://upload.wikimedia.org/wikipedia/commons/thumb/a/ac/No_image_available.svg/480px-No_image_available.svg.png"
  items$Photo <- paste0('')
  datatable(items[, c("Photo", "title", "category",
"condition", "pickup_location")], escape = FALSE, options =
list(pageLength = input$rows_display))
})

output$requests_table <- renderDT({
  query <- "SELECT request_id, user_id AS requester_id, item_id,
status, request_date FROM requests"
  datatable(dbGetQuery(con, query), options = list(pageLength =
input$rows_display))
})

transactions_data <- reactive({
  query <- "SELECT transaction_id, request_id, donor_id, status,
date_time FROM transactions"
  df <- dbGetQuery(con, query)
  df$date_time <- as.character(df$date_time)
  df
})

output$transactions_table <- renderDT({
  datatable(transactions_data(), options = list(pageLength =
input$rows_display))
})

selected_transaction <- reactive({
  req(input$transactions_table_rows_selected)
  transactions_data()[input$transactions_table_rows_selected, ]
})

```

```

output$download_receipt <- downloadHandler(
  filename = function() paste0("Donation_Receipt_", Sys.Date(),
".txt"),
  content = function(file) {
    txn <- selected_transaction()
    receipt <- c(
      "CAMPUSGIVES DONATION RECEIPT",
      "-----",
      paste("Transaction ID:", txn$transaction_id),
      paste("Request ID:", txn$request_id),
      paste("Donor ID:", txn$donor_id),
      paste("Status:", txn$status),
      paste("Date:", txn$date_time),
      "\nThank you for your generous donation!"
    )
    writeLines(receipt, file)
  }
)

output$pickup_map <- renderLeaflet({
  items <- get_items()
  if (nrow(items) == 0) return(leaflet() %>% addTiles())
  locations <- data.frame(
    lat = runif(nrow(items), 32.98, 32.99),
    lng = runif(nrow(items), -96.75, -96.74),
    label = items$title,
    link = paste0("https://www.google.com/maps?q=",
runif(nrow(items), 32.98, 32.99), ",", runif(nrow(items), -96.75,
-96.74))
  )
  leaflet(locations) %>% addTiles() %>% addMarkers(~lng, ~lat,
popup = ~paste0("<b>", label, "</b><br><a href='", link, "'
target='_blank'>Get Directions</a>"))
})

output$stats_plot <- renderPlot({
  choice <- input$stat_choice
  if (choice == "Transactions Over Time") {
    data <- dbGetQuery(con, "SELECT date_time FROM
transactions")
    data$date_time <- as.Date(data$date_time)
    ggplot(data, aes(date_time)) + geom_histogram(binwidth = 1,
fill = "#0072B2", color = "white") + labs(title = "Transactions
Over Time", x = "Date", y = "Count") + theme_minimal()
  } else if (choice == "Transactions by Status") {
    data <- dbGetQuery(con, "SELECT status FROM transactions")
  }
})

```

```
      ggplot(data, aes(status)) + geom_bar(fill = "#E69F00") +
labs(title = "By Status", x = "Status", y = "Count") +
theme_minimal()
    } else if (choice == "Requests per Day") {
      data <- dbGetQuery(con, "SELECT request_date FROM requests")
      data$request_date <- as.Date(data$request_date)
      ggplot(data, aes(request_date)) + geom_bar(fill = "#009E73")
+ labs(title = "Requests Per Day", x = "Date", y = "Count") +
theme_minimal()
    } else {
      data <- get_items()
      data$category <- str_wrap(data$category, width = 15)
      ggplot(data, aes(x = category)) + geom_bar(fill = "#D55E00")
+ labs(title = "Available Items", x = "Category", y = "Count") +
theme_minimal() + theme(axis.text.x = element_text(angle = 45,
hjust = 1))
    }
  })
}

# Launch App
shinyApp(ui, server)
```