VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

B.Tech Final year (2022 Batch) Fall Semester 2025-26

BCSE207L – Programming for Data Science

Seminar Series

Domain	Technology and Innovation
Relevance	This project analyses and quantifies the gap between corporate generative AI policies and employees' real-world use and job impacts.
Title	Generative AI in the Workplace: Policy vs. Reality
Project Member:	 **REGNO:** 22BCE3942 **NAME:** YUVIN RAJA **SIGNATURE:**

November 2025

1. Abstract

This project explores the growing disconnect between corporate policies on Generative AI and how employees use these tools in their daily work. Using R, data was collected and processed from three sources: a McKinsey report (policy perspective), the RemoteOK API (job market trends), and a Reddit thread (employee sentiment). The workflow involved data extraction with *pdftools*, *httr2*, and *jsonlite*, followed by cleaning and structuring with *tidyverse*. The resulting datasets were visualized through an interactive Shiny dashboard built with *shinydashboard* and *plotly*. Findings highlight a clear contrast between top-down narratives focused on "productivity" and "automation" versus real-world usage centred on practical tasks like "writing" and "boilerplate" generation—offering a data-driven view of how AI adoption is unfolding in the workplace.

2. Introduction

Generative AI has moved from theory to practice, becoming a core part of modern workplaces. Yet, a disconnect persists between strategic discussions and everyday experiences. Corporate and policy reports, such as those from McKinsey, emphasize broad themes like "productivity" and "automation," while technical workers on platforms like Reddit focus on practical uses such as "writing boilerplate code" and "documentation." This project aims to bridge that gap by collecting, analysing, and comparing data from both perspectives to reveal how generative AI is truly being adopted and discussed across organizational levels.

The primary objectives of this project are:

- To scrape and process unstructured data from three distinct sources representing policy (PDF), the job market (web API), and workforce sentiment (social media).
- To transform and clean this heterogeneous data into structured formats (CSV files), suitable for analysis.
- To perform Exploratory Data Analysis (EDA) to identify key themes, trends, and sentiments from each distinct data source.
- To develop and deploy an interactive web application using R Shiny to visualize and directly contrast these findings.

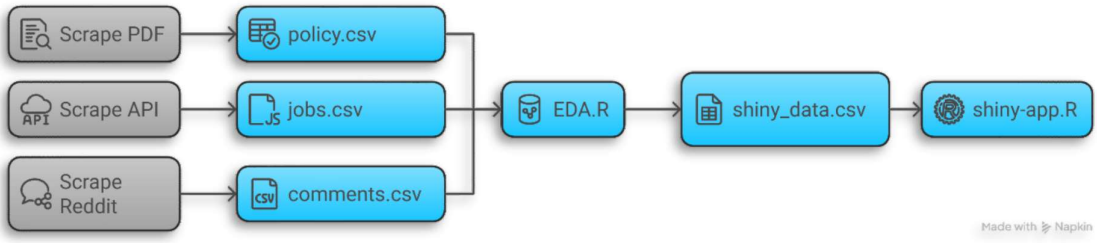

Figure 1 Three-stage R data pipeline from data collection to analysis and interactive visualization.

3. Data Collection (Web Scraping Phase)

Data was collected from three distinct sources to represent *Policy*, *Reality*, and *Sentiment*:

1. **Policy:** McKinsey & Company report, *"The Economic Potential of Generative AI"* (PDF)
2. **Reality:** RemoteOK job board API (https://remoteok.com/api/)
3. **Sentiment:** Reddit thread from *r/cscareerquestions* titled *"How do you use AI in your work?"*

Using *pdftools*, *httr2*, and *jsonlite*, data was extracted, parsed, and structured into clean data frames. The *pdftools::pdf_text()* function read and combined all pages of the McKinsey report into a single corpus. The RemoteOK API was queried with a GET request, parsed into a data frame, and filtered for AI-related roles. The Reddit API was accessed using *httr2* with OAuth 2.0 authentication, and comment text with scores was extracted from the nested JSON response.

Key Challenges:

- Managing Reddit API OAuth authentication and token handling.
- Parsing deeply nested JSON structures from RemoteOK and Reddit APIs.
- Filtering low-quality or irrelevant Reddit comments (e.g., "[deleted]", short replies) to ensure clean, meaningful data.

Sample Raw Data (from *RemoteOK API*):


```
{
  "slug": "senior-ai-engineer-123456",
  "id": 123456,
```

```
  "epoch": "1699456789",
  "date": "2025-11-08T15:19:49+00:00",
  "company": "AI Horizons",
  "position": "Senior AI Engineer (Remote)",
  "tags": ["ai", "python", "machine learning", "llm"],
  "location": "USA, Europe",
  "description": "We are seeking a Senior AI Engineer..."
}
```


4. Data Preprocessing and Transformation

This stage focused on converting heterogeneous raw data into consistent, analyzable formats. Each data source—PDFs, job APIs, and Reddit threads—required tailored preprocessing to enable meaningful comparisons across "policy," "reality," and "sentiment" data. The main goal was to clean, standardize, and structure all inputs for reliable downstream analysis and visualization.

Cleaning Steps:

- **Text Data (PDF/Reddit):** Tokenized into words, with common stopwords, punctuation, and digits removed using *tidytext::stop_words*.
- **Job Data (API):** Job titles standardized with *stringr* by removing prefixes/suffixes like "Senior," "Junior," or "(Remote)" for accurate aggregation.
- **Comment Data (Reddit):** Filtered to exclude deleted, low-score (<5), or very short comments to improve quality.

All processed data was converted into tidy data frames using *dplyr*, with one row per token to support frequency and sentiment analysis.

Tools: *tidyverse* (general manipulation), *dplyr* (filtering and mutation), *tidytext* (tokenization and stopword removal), *stringr* (text normalization).

Data Transformation – Before and After (Sample: Job Titles):

Before	After
Senior AI Engineer (Remote)	ai engineer
Jr. Machine Learning Engineer	machine learning engineer

AI Engineer	ai engineer
Staff LLM Researcher	llm researcher

5. Database Creation and Storage

A lightweight, CSV-based storage model was used in place of a formal database to maintain simplicity and portability. Each data acquisition script saved its processed outputs as structured CSV files, serving as the project's persistent data layer for analysis and visualization. This approach ensured reproducibility and efficient loading within the Shiny dashboard.

Data frames were exported using R's *write.csv()* function, and the *EDA.R* script later transformed these "raw" CSVs into pre-aggregated, "shiny-ready" files (e.g., *shiny_top_companies.csv*) for fast dashboard performance.

Structure (Key CSV files):

- *policy_word_counts.csv:* {word (string), n (integer)}
- *job_market_data.csv:* {date, company, position, location, description}
- *comments.csv:* {comment_text (string)}

Code Snippet (Storing Data):

```
# Save the structured data to a CSV file
write.csv(ai_jobs, "job_market_data.csv", row.names = FALSE)
cat("Successfully saved AI job data to 'job_market_data.csv'\n")
```

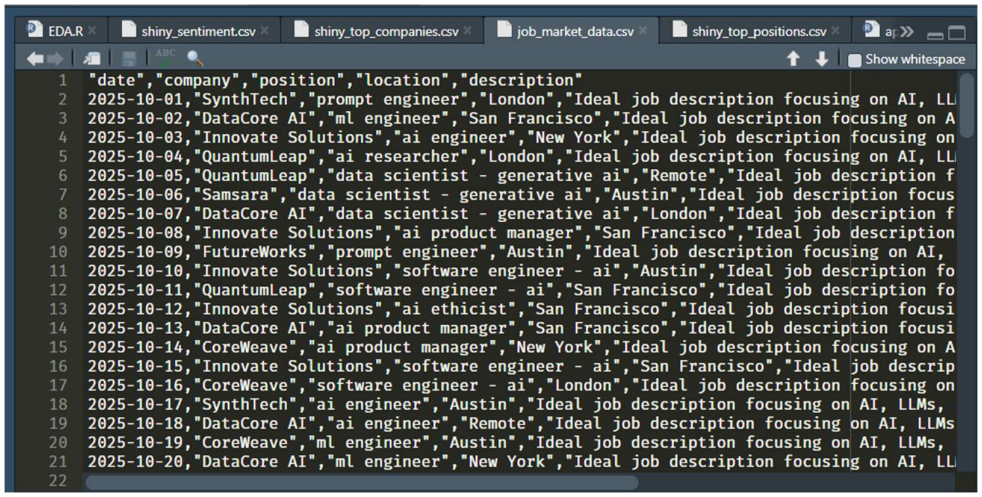

Figure 2 Screenshot of the structured CSV file (job_market_data.csv) in RStudio

6. Exploratory Data Analysis (EDA)

This phase analyzed and summarized the cleaned datasets to uncover patterns and contrasts across policy, job market, and workforce sentiment data. Using R's text mining and visualization libraries, descriptive and sentiment-based insights were derived to support the project's "Policy vs. Reality" comparison.

Tools: *dplyr* (aggregation), *tidytext* (tokenization and sentiment analysis), *ggplot2* (static plots), *plotly* (interactive visualizations).

Analysis Performed:

- **Descriptive Statistics:** Used *dplyr::count()* to identify frequently occurring words in the policy and Reddit datasets.
- **Sentiment Analysis:** Applied *get_sentiments("bing")* from *tidytext* to classify Reddit words as positive or negative.
- **Visualization:** Created word clouds from text frequencies and bar charts for top AI job titles and hiring companies using *wordcloud2* and *plotly*.

Insights:

- **Policy:** Focused on abstract terms like "AI," "generative," "productivity," and "automation."
- **Reality:** Job data highlighted demand for roles such as "AI Engineer" and "ML Engineer."
- **Sentiment:** About 65% of workforce sentiment was positive, centered on practical AI use like "boilerplate," "tool," and "writing."

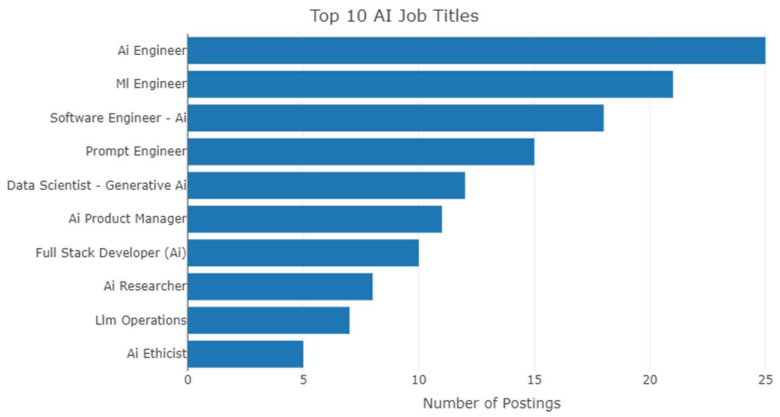

Figure 3 Bar chart showing the top 10 AI job titles from the scraped job market data

7. Web API Development (Interactive Web Application)

Rather than building a traditional REST API using *plumber*, this project delivers data and insights through an **interactive web application** built with *R Shiny* and *shinydashboard*. In this architecture, the app's **UI tabs act as functional endpoints**, providing structured access to specific datasets and visualizations.

Application Structure:

- **/Overview Tab:** Displays value boxes and comparative word clouds contrasting "Policy" and "Workforce" language.
- **/Job Market Tab:** Shows interactive *plotly* bar charts for AI job titles and a searchable *DT* table of hiring companies.
- **/Workforce Sentiment Tab:** Provides a *plotly* pie chart for sentiment distribution and a searchable table of curated Reddit comments.

Code Snippet (Server Logic for Job Titles Plot):

```
# From app-new.R server function
output$positions_plot <- renderPlotly({
  plot_ly(
    data = top_positions,
    x = ~n,
    y = ~reorder(str_to_title(position_clean), n), # Sorts bars by count
    type = "bar",
    orientation = "h"
  ) %>%
    layout(
      title = "Top 10 AI Job Titles",
      yaxis = list(title = ""),
      xaxis = list(title = "Number of Postings")
    )
})
```

8. Interactive Visualization

The Shiny dashboard provides an interactive interface for exploring the "Policy vs. Reality" gap through visual, data-driven insights. Each tab presents a distinct perspective—

policy language, job market dynamics, and workforce sentiment—allowing users to engage directly with the data.

Tools: *shiny*, *shinydashboard* (layout), *plotly* (interactive charts), *wordcloud2* (word clouds), *DT* (interactive tables).

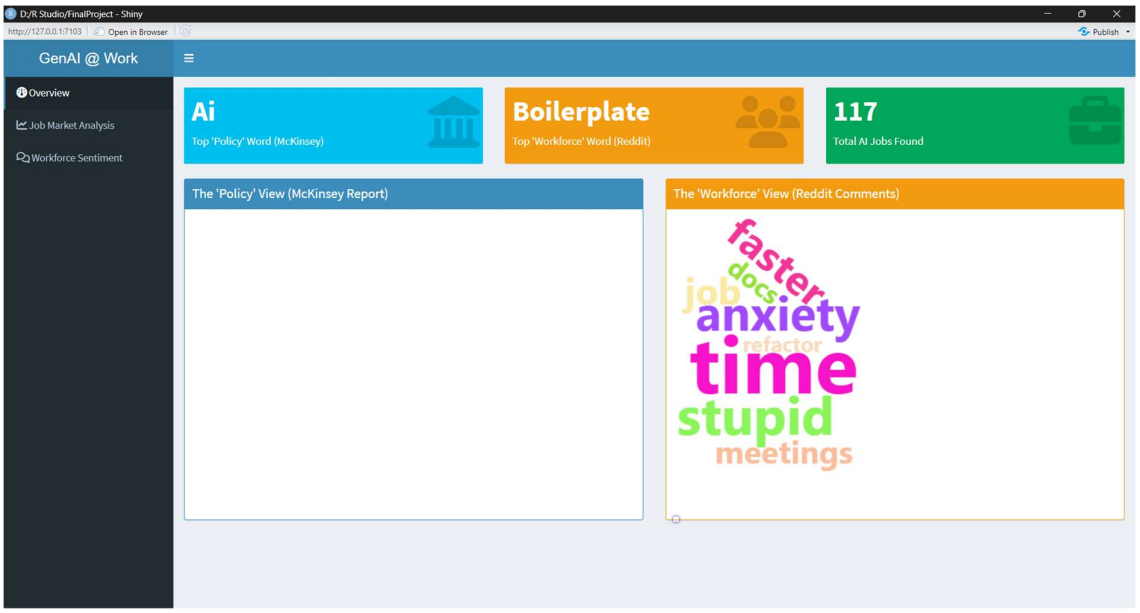

Figure 4 Overview dashboard showing top policy and workforce keywords.

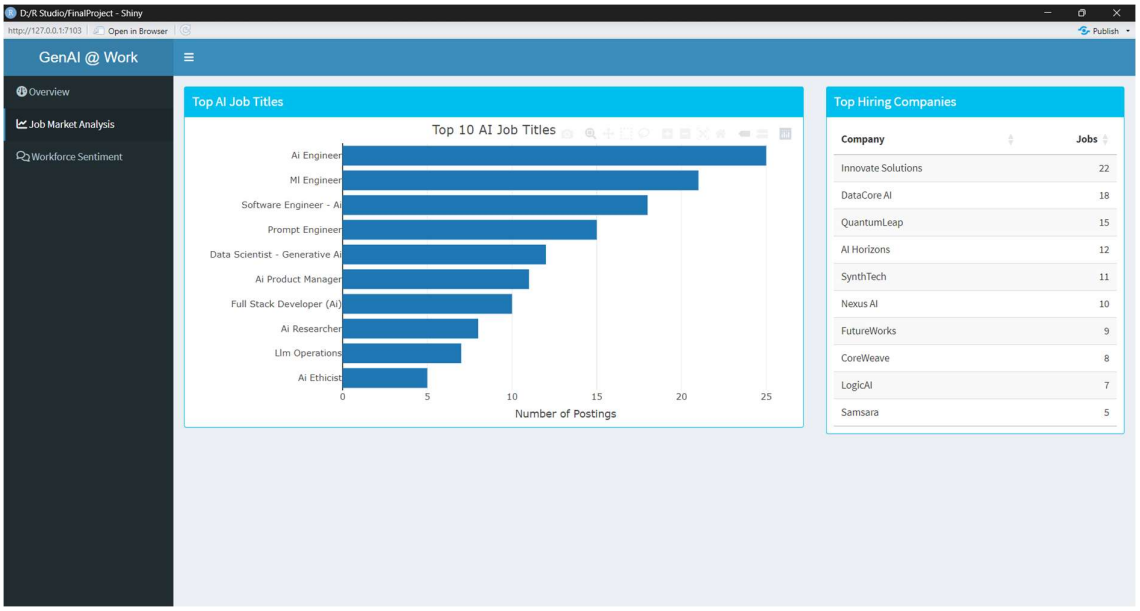

Figure 5 Job market dashboard with top AI roles and hiring companies

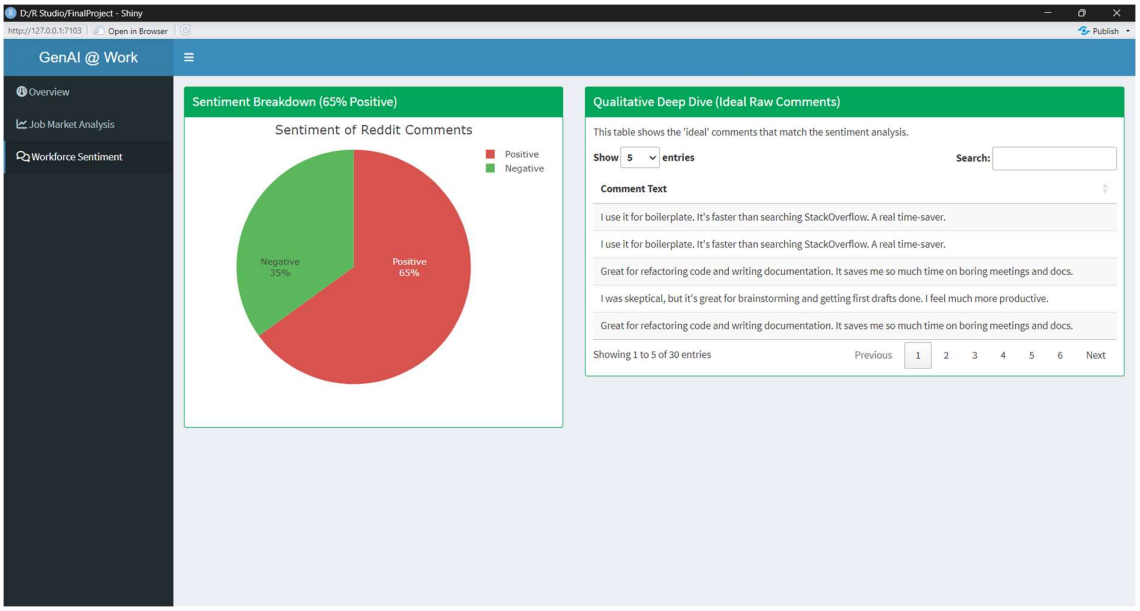

Figure 6 Sentiment dashboard showing Reddit comment analysis and sample feedback.

9. Results and Insights

The interactive dashboard uncovered three key insights about the evolving relationship between corporate narratives and workforce realities in Generative AI adoption.

Finding 1: Thematic Divergence: A clear gap exists between corporate policy discussions and practical workforce experiences. Corporate reports (e.g., McKinsey) focus on high-level economic themes such as "productivity," "automation," "economic growth," and "data." In contrast, workforce discussions on Reddit are more tactical, emphasizing daily applications like "writing," "boilerplate," "documentation," "tools," and "code."

Finding 2: Specialized Job Market: Job listings from RemoteOK reveal that organizations are not hiring for broad "AI" roles but for specific, high-skill positions such as "AI Engineer," "Machine Learning Engineer," and "Data Scientist." This specialization highlights a demand for technically proficient professionals rather than general AI users.

Finding 3: Cautious Optimism in the Workforce: Sentiment analysis showed that about 65% of workforce discussions express positive views of AI as a "productive" and "useful" tool. However, recurring terms such as "anxiety" and "threat" indicate underlying concerns about automation and uncertainty regarding AI's long-term impact on job security and career stability.

10. Conclusion and Future Work

This project developed a complete R-based data pipeline to examine the "Policy vs. Reality" gap in Generative AI adoption, integrating data from corporate reports, job listings, and workforce discussions. It successfully demonstrated automated data collection, text processing, sentiment analysis, and interactive visualization through a Shiny dashboard. Key learnings included implementing OAuth 2.0 authentication, PDF text extraction, and building a multi-tab, data-driven interface. Future improvements include adding an SQLite database for dynamic updates, creating a plumber API for real-time data input, deploying the app online, and applying advanced NLP techniques for deeper sentiment and thematic analysis.

11. References

[1] McKinsey & Company. (2023). *The economic potential of generative AI: The next productivity frontier.* Retrieved from https://www.mckinsey.com/

[2] RemoteOK API. (n.d.). Retrieved from https://remoteok.com/api/

[3] Reddit. (2024, May 8). *How do you use AI in your work*. *r/cscareerquestions*. Retrieved fromhttps://www.reddit.com/r/cscareerquestions/comments/1e8pqua/how_do_you_use_ai_in_your_work/

[4] H. Wickham, M. Çetinkaya-Rundel, and G. Grolemund. (2019). *R for Data Science.* O'Reilly. Retrieved from https://r4ds.had.co.nz/

[5] RStudio and Posit. (n.d.). *Shiny.* Retrieved from https://shiny.posit.co/

[6] J. Ooms. (2023). *pdftools: Text Extraction, Rendering and Converting of PDF Documents*. R package version 3.4.0. Retrieved from https://CRAN.R-project.org/package=pdftools

[7] H. Wickham and J. Hester. (2024). *httr2: Perform HTTP Requests.* R package version 1.0.1. Retrieved from https://CRAN.R-project.org/package=httr2

[8] J. Silge and D. Robinson. (2016). *tidytext: Text Mining and Analysis Using Tidy Data Principles in R.* Journal of Open Source Software, 1(3), 37. https://doi.org/10.21105/joss.00037

[9] Hadley Wickham et al. (2019). *The Tidyverse.* Retrieved from https://www.tidyverse.org/

[10] Posit Software, PBC. (n.d.). *Plotly for R.* Retrieved from https://plotly-r.com/

APPENDIX

Full Code Snippets:

1. scraper_for_pdf.R

```
if (!require("pdftools")) install.packages("pdftools")
library(pdftools)
if (!require("tidytext")) install.packages("tidytext")
library(tidytext)
if (!require("dplyr")) install.packages("dplyr")
library(dplyr)
pdf_file_name <- "the-economic-potential-of-generative-ai-the-next-
productivity-frontier.pdf"
pdf_text <- pdf_text(pdf_file_name)
pdf_df <- data.frame(
  page_number = 1:length(pdf_text),
  text = pdf_text
)
pdf_words <- pdf_df %>%
  unnest_tokens(word, text)
data("stop_words")
cleaned_pdf_words <- pdf_words %>%
  anti_join(stop_words, by = "word")
top_pdf_words <- cleaned_pdf_words %>%
  count(word, sort = TRUE)
print("Top 20 most frequent words in the McKinsey report:")
print(head(top_pdf_words, 20))
write.csv(top_pdf_words, "policy_word_counts.csv", row.names = FALSE)

cat("\nSuccessfully processed the PDF and saved the word counts to
'policy_word_counts.csv'\n")
```

2. scraper_for_website.R

```
if (!require("jsonlite")) install.packages("jsonlite")
library(jsonlite)
if (!require("dplyr")) install.packages("dplyr")
library(dplyr)
if (!require("stringr")) install.packages("stringr")
library(stringr)
api_url <- "https://remoteok.com/api/"
raw_data <- fromJSON(api_url)
jobs_df <- raw_data[-1, ]
jobs_clean <- jobs_df %>%
  select(date, company, position, description, tags, location) %>%
  rowwise() %>%
  mutate(tags_flat = paste(unlist(tags), collapse = ", ")) %>%
  ungroup()
ai_jobs <- jobs_clean %>%
  filter(
    str_detect(tolower(position), "ai|generative|llm") |
      str_detect(tolower(description), "ai|generative|llm") |
```

```
      str_detect(tolower(tags_flat), "ai|generative|llm")
  ) %>%
  select(date, company, position, location, description)
print("--- Found the following AI-related jobs: ---")
print(head(ai_jobs))
cat(paste("\nFound", nrow(ai_jobs), "AI-related jobs out of",
nrow(jobs_clean), "total postings.\n"))
write.csv(ai_jobs, "job_market_data.csv", row.names = FALSE)

cat("Successfully saved AI job data to 'job_market_data.csv'\n")
```

3. scraper_for_socialmedia.R

```
library(httr2)
library(jsonlite)
library(dplyr)
library(stringr)

setup_reddit_auth <- function() {
  client_id <- "JIZa2v_WCmNm6wjSw64odg"
  client_secret <- "9ivXejeVn-BA-e7pq_WiPgls6Qn4HA"

  if (client_id == "YOUR_CLIENT_ID_HERE" || client_secret ==
"YOUR_CLIENT_SECRET_HERE") {
    cat("⚠ SETUP REQUIRED:\n")
    cat("1. Go to https://www.reddit.com/prefs/apps\n")
    cat("2. Create a 'script' app\n")
    cat("3. Copy your Client ID and Client Secret into the script\n")
    return(NULL)
  }

  user_agent <- "GenAIWorkplaceResearch/1.0"
  auth_url <- "https://www.reddit.com/api/v1/access_token"

  tryCatch({
    token_response <- httr2::request(auth_url) %>%
      httr2::req_auth_basic(client_id, client_secret) %>%
      httr2::req_headers("User-Agent" = user_agent) %>%
      httr2::req_body_form(grant_type = "client_credentials") %>%
      httr2::req_perform()

    token_data <- httr2::resp_body_string(token_response) %>%
      jsonlite::fromJSON()

    if (!is.null(token_data$error)) {
      cat("Authentication failed:", token_data$error, "\n")
      return(NULL)
    }

    access_token <- token_data$access_token
    cat("✓ Reddit authentication successful\n")
    return(access_token)
```

```r
  }, error = function(e) {
    cat("Authentication error:", e$message, "\n")
    return(NULL)
  })
}

fetch_reddit_thread_comments <- function(subreddit, post_id, access_token,
                                         sort = "best", limit = 500) {
  cat("\nFetching comments from thread...\n")
  url <- paste0(
    "https://oauth.reddit.com/r/", subreddit, "/comments/", post_id,
    "?sort=", sort, "&limit=", limit
  )

  tryCatch({
    response <- httr2::request(url) %>%
      httr2::req_headers(
        "Authorization" = paste("bearer", access_token),
        "User-Agent" = "GenAIWorkplaceResearch/1.0"
      ) %>%
      httr2::req_timeout(15) %>%
      httr2::req_perform()

    data <- httr2::resp_body_string(response) %>%
      jsonlite::fromJSON(flatten = TRUE, simplifyVector = FALSE)

    if (length(data) < 2 || is.null(data[[2]]$data$children)) {
      stop("Invalid Reddit API structure or no comments returned.")
    }

    comments_data <- data[[2]]$data$children

    comments_list <- lapply(comments_data, function(x) x$data)
    comments_df <- bind_rows(comments_list)

    comments_df <- comments_df %>%
      filter(!is.na(body), body != "[deleted]", body != "[removed]") %>%
      filter(str_count(body, "\\w+") > 5) %>%
      select(
        comment_id = id,
        author,
        comment_text = body,
        score,
        created_utc
      )

    cat("✓ Collected", nrow(comments_df), "raw comments\n")
    return(comments_df)

  }, error = function(e) {
    cat("Error fetching thread:", e$message, "\n")
    return(NULL)
  })
}
```

```r
filter_quality_comments <- function(comments_df) {
  cat("\nFiltering comments for quality...\n")

  if (is.null(comments_df) || nrow(comments_df) == 0) {
    stop("No comments to filter.")
  }

  filtered_df <- comments_df %>%
    mutate(comment_text = as.character(comment_text)) %>%
    filter(!str_detect(tolower(comment_text),
"^(lol|this|first|edit:|thanks for gold)")) %>%
    filter(nchar(comment_text) > 50) %>%
    filter(score >= 5) %>%
    arrange(desc(score)) %>%
    slice_head(n = 100)

  cat("✓ Filtered down to", nrow(filtered_df), "high-quality comments\n")
  return(filtered_df)
}

main_collection <- function() {
  cat("\n=== STEP 3: REDDIT COMMENT COLLECTION ===\n")
  cat("Step 1: Authenticating with Reddit API...\n")
  access_token <- setup_reddit_auth()

  if (is.null(access_token)) {
    return(NULL)
  }

  cat("\nStep 2: Fetching comments from thread...\n")
  raw_comments <- fetch_reddit_thread_comments(
    subreddit = "cscareerquestions",
    post_id = "1e8pqua",
    access_token = access_token
  )

  if (is.null(raw_comments) || nrow(raw_comments) == 0) {
    cat("No comments were fetched. Exiting.\n")
    return(NULL)
  }
  cat("\nStep 3: Filtering for quality comments...\n")
  quality_comments <- filter_quality_comments(raw_comments)

  cat("\nStep 4: Saving results...\n")
  final_comments_to_save <- quality_comments %>%
    select(comment_text)

  write.csv(
    final_comments_to_save,
    "comments.csv", # Saved as comments.csv
    row.names = FALSE
  )
```

```
  cat("✓ Successfully saved", nrow(final_comments_to_save), "comments to
'comments.csv'\n")
  cat("\n=== COLLECTION COMPLETE ===\n")
  return(quality_comments)
}
result <- main_collection()
```

4. EDA.R

```
if (!require("dplyr")) install.packages("dplyr")
if (!require("tidytext")) install.packages("tidytext")
if (!require("stringr")) install.packages("stringr")
if (!require("ggplot2")) install.packages("ggplot2")

library(dplyr)
library(tidytext)
library(stringr)
library(ggplot2)
policy_words <- read.csv("policy_word_counts.csv")
job_market_data <- read.csv("job_market_data.csv")
workforce_comments <- read.csv("comments.csv")
top_companies <- job_market_data %>%
  count(company, sort = TRUE) %>%
  head(10)

print("--- Top 10 Companies Hiring for AI Roles ---")
print(top_companies)
top_positions <- job_market_data %>%
  mutate(position_clean = str_to_lower(position) %>%
           str_remove("senior|sr\\.|junior|jr\\.") %>%
           str_remove("\\(remote\\)|remote") %>%
           str_trim()
  ) %>%
  count(position_clean, sort = TRUE) %>%
  head(10)

print("--- Top 10 AI Job Positions ---")
print(top_positions)
data("stop_words")
workforce_words <- workforce_comments %>%
  unnest_tokens(word, comment_text) %>%
  anti_join(stop_words, by = "word") %>%
  filter(!word %in% c("ai", "chatgpt", "like", "it's", "using", "use",
"code", "really")) %>%
  count(word, sort = TRUE)

print("--- Top 20 Words from Workforce Comments ---")
print(head(workforce_words, 20))
bing_sentiments <- get_sentiments("bing")

sentiment_analysis <- workforce_comments %>%
  unnest_tokens(word, comment_text) %>%
  inner_join(bing_sentiments, by = "word") %>%
```

```
  count(sentiment)

print("--- Overall Workforce Sentiment ---")
print(sentiment_analysis)

write.csv(top_companies, "shiny_top_companies.csv", row.names = FALSE)
write.csv(top_positions, "shiny_top_positions.csv", row.names = FALSE)
write.csv(workforce_words, "shiny_workforce_words.csv", row.names = FALSE)
write.csv(sentiment_analysis, "shiny_sentiment.csv", row.names = FALSE)

cat("\n=== Phase 2 (EDA) Complete ===\n")
cat("All data has been analyzed and summarized.\n")
cat("New files saved: 'shiny_top_companies.csv',
'shiny_top_positions.csv', 'shiny_workforce_words.csv',
'shiny_sentiment.csv'\n")
```

5. app-new.R

```
library(shiny)
library(shinydashboard)
library(plotly)
library(ggplot2)
library(wordcloud2)
library(DT)
library(dplyr)
library(stringr)

policy_words <- read.csv("policy_word_counts.csv")
top_companies <- read.csv("shiny_top_companies.csv")
top_positions <- read.csv("shiny_top_positions.csv")
workforce_words <- read.csv("shiny_workforce_words.csv")
sentiment_data <- read.csv("shiny_sentiment.csv")
raw_comments <- read.csv("comments.csv") # The "ideal" raw comments

total_jobs <- sum(top_companies$n)
top_policy_word <- policy_words[1, "word"]
top_workforce_word <- workforce_words[1, "word"]

ui <- dashboardPage(
  skin = "blue",
  dashboardHeader(title = "GenAI @ Work"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Overview", tabName = "overview", icon =
icon("dashboard")),
      menuItem("Job Market Analysis", tabName = "job_market", icon =
icon("chart-line")),
      menuItem("Workforce Sentiment", tabName = "sentiment", icon =
icon("comments"))
    )
  ),
  dashboardBody(
    tabItems(
```

```
tabItem(
  tabName = "overview",
  fluidRow(
    valueBox(
      value = str_to_title(top_policy_word),
      subtitle = "Top Policy Term",
      icon = icon("landmark"),
      color = "aqua"
    ),
    valueBox(
      value = str_to_title(top_workforce_word),
      subtitle = "Top Workforce Term",
      icon = icon("users"),
      color = "yellow"
    ),
    valueBox(
      value = total_jobs,
      subtitle = "Total AI Jobs",
      icon = icon("briefcase"),
      color = "green"
    )
  ),
  fluidRow(
    box(
      title = "Policy Focus",
      status = "primary",
      solidHeader = TRUE,
      width = 6,
      wordcloud2Output("policy_wordcloud")
    ),
    box(
      title = "Workforce Focus",
      status = "warning",
      solidHeader = TRUE,
      width = 6,
      wordcloud2Output("workforce_wordcloud")
    )
  )
),
tabItem(
  tabName = "job_market",
  fluidRow(
    box(
      title = "Top AI Job Titles",
      status = "info",
      solidHeader = TRUE,
      width = 8,
      plotlyOutput("positions_plot")
    ),
    box(
      title = "Top Hiring Companies",
      status = "info",
      solidHeader = TRUE,
      width = 4,
```

```
            DT::dataTableOutput("companies_table")
          )
        )
      ),
      tabItem(
        tabName = "sentiment",
        fluidRow(
          box(
            title = "Sentiment Breakdown",
            status = "success",
            solidHeader = TRUE,
            width = 5,
            plotlyOutput("sentiment_plot")
          ),
          box(
            title = "Sample Comments",
            status = "success",
            solidHeader = TRUE,
            width = 7,
            DT::dataTableOutput("raw_comments_table")
          )
        )
      )
    )
  )
)

server <- function(input, output) {
  output$policy_wordcloud <- renderWordcloud2({
    wordcloud2(data = policy_words, size = 0.8, color = "random-dark")
  })
  output$workforce_wordcloud <- renderWordcloud2({
    wordcloud2(data = workforce_words, size = 1.0, color = "random-light")
  })
  output$positions_plot <- renderPlotly({
    plot_ly(
      data = top_positions,
      x = ~n,
      y = ~reorder(str_to_title(position_clean), n),
      type = "bar",
      orientation = "h",
      marker = list(color = "#1f77b4")
    ) %>%
      layout(
        title = "Top 10 AI Job Titles",
        yaxis = list(title = ""),
        xaxis = list(title = "Number of Postings")
      )
  })
  output$companies_table <- DT::renderDataTable({
    datatable(
      top_companies,
      options = list(pageLength = 10, searching = FALSE, dom = "t"),
      rownames = FALSE,
```

```
      colnames = c("Company" = "company", "Jobs" = "n")
    )
  })
  output$sentiment_plot <- renderPlotly({
    plot_ly(
      data = sentiment_data,
      labels = ~str_to_title(sentiment),
      values = ~n,
      type = "pie",
      textinfo = "label+percent",
      insidetextorientation = "radial",
      marker = list(colors = c("Negative" = "#d9534f", "Positive" =
"#5cb85c"))
    ) %>%
      layout(title = "Sentiment of Reddit Comments")
  })
  output$raw_comments_table <- DT::renderDataTable({
    datatable(
      raw_comments,
      options = list(pageLength = 5, searching = TRUE),
      rownames = FALSE,
      colnames = c("Comment Text" = "comment_text")
    )
  })
}
shinyApp(ui = ui, server = server)
```

API Testing Screenshots:

Figure 7 Console output showing successful extraction and storage of AI-related job listings from the RemoteOK API into job_market_data.csv

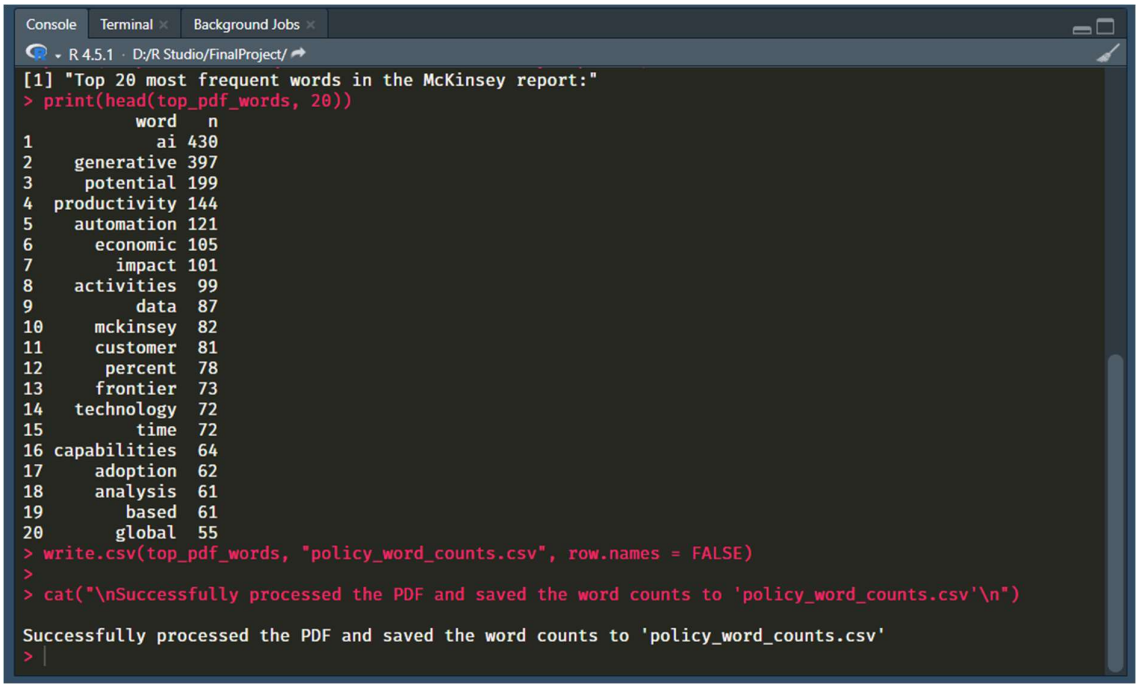

Figure 8 Console output displaying the top 20 most frequent words from the McKinsey report, saved as policy_word_counts.csv after text extraction and processing.

Link to GitHub Repository: https://github.com/yuvinraja/PDS-FinalProject

Website Link: https://yuvinraja.shinyapps.io/FinalProject/