ALY 6070: Communication and Visualization for Data Analytics

Assignment 2: Northeastern University Job Application for Teaching Assistant

Submitted to

Prof. Fatemeh Ahmadi Abkenari

Submitted by

Abhilash Dikshit

Milan Prajapati

Shamim Sherafati

Smit Parmar

College of Professional Studies
Northeastern University
Vancouver, Canada

Assignment 2: Northeastern University Job Application for Teaching Assistant

College of Professional Studies
Northeastern University
Vancouver, Canada

Abstract

This R code creates a web-based form for job application as a Teaching Assistant at Northeastern University. The form includes various fields such as personal details, education level, teaching experience, working knowledge in programming languages, availability, interview availability, status in Canada, salary expectation, and file upload for resume. The data is collected through this form and is processed in R for data analysis.

Introduction

The purpose of this code is to create a job application form for the position of Teaching Assistant at Northeastern University. The form is created using the Shiny package in R, which allows for the creation of interactive web applications. The form includes various fields for the candidate to fill out, such as their name, gender, address, education level, teaching experience, working knowledge in R and/or Python, availability, interview availability, status in Canada, expected salary, and a field to upload their resume.

The code begins by loading the necessary libraries for the application, such as plyr, plotly, ggplot2, psych, tidyr, dplyr, lubridate, readr, caret, caTools, glmnet, shiny, shinyvalidate, shinyWidgets, shinythemes, tippy, and shinyjs. These libraries are used to provide various functionalities such as data wrangling, data visualization, machine learning, and web development. The libraries are loaded using the lapply() function, which ensures that all packages are installed and loaded before running the application.

Next, the code sets up the user interface (UI) for the application using the fluidPage() function from the shiny package. The UI is designed using the lumen theme from the shinythemes package and includes various input fields for the candidate to fill out. The background image of the UI is set to the logo of Northeastern University, which is positioned to the right of the screen. The sidebar panel of the UI includes fields for the candidate's personal information such as their name, gender, address, email, and phone number. The main panel of the UI includes fields for the candidate's education level, teaching experience, working knowledge in R and/or Python, availability, interview availability, status in Canada, expected salary, and a field to upload their resume. The candidate is also required to agree to the terms and conditions before submitting the application.

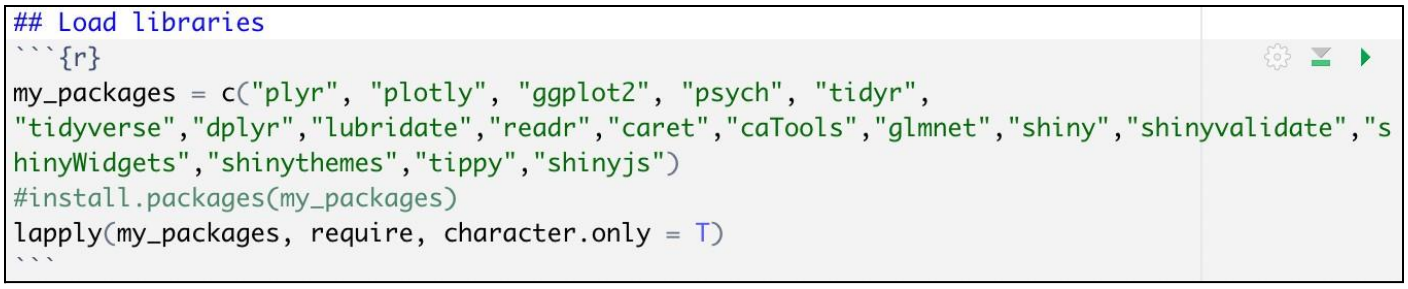
The code then sets up the server logic for the application using the server() function from the shiny package. The server logic includes various input validation and error handling functions to ensure that the candidate enters the correct information in the input fields. For example, the observeEvent() function is used to check if the candidate has entered their first name and last name. If either of these fields is empty, an error message is displayed.

Problem:

Build an application representing an online form for a person to apply for a job with Shiny R. This form must have different widgets (at least four). For example a select box of Canadian states, a radio button of the selected job the person is trying to apply.

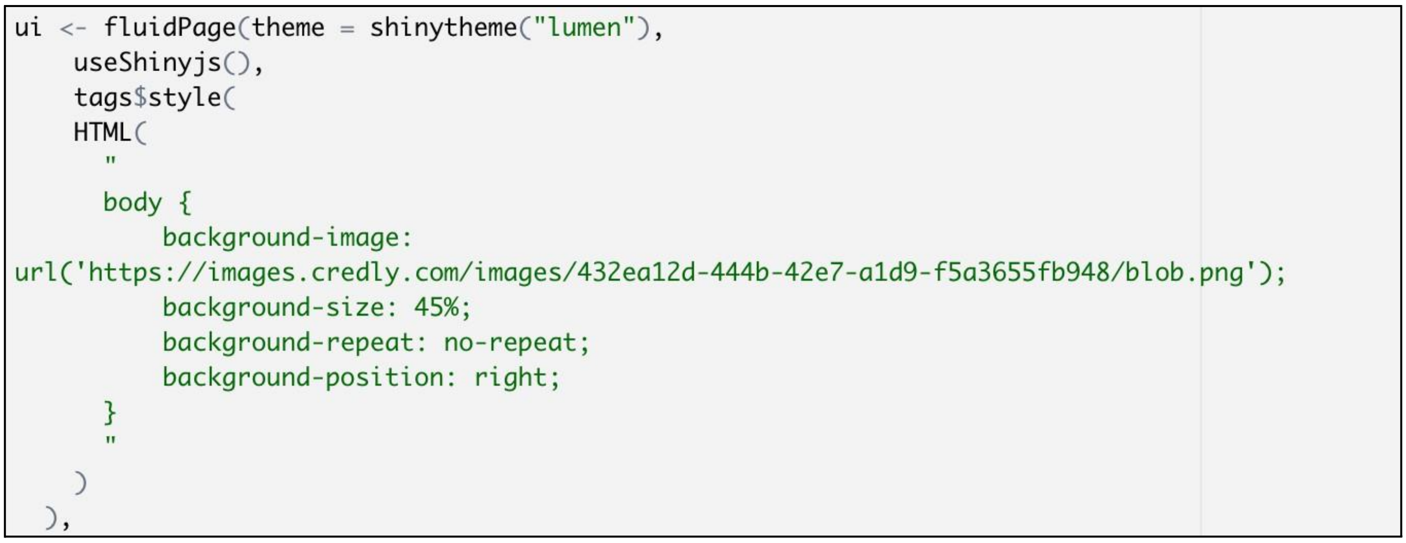
Loading the Libraries

For the first step, we load the required libraries that we shall be using during the code. The libraries that are used in this project are shown below.



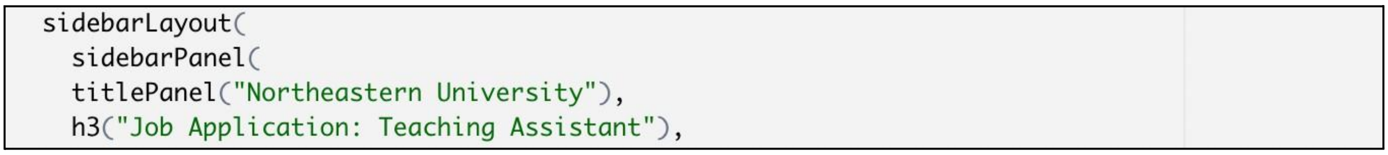
~~~
## Load libraries
```{r}
my_packages = c("plyr", "plotly", "ggplot2", "psych", "tidyr",
"tidyverse","dplyr","lubridate","readr","caret","caTools","glmnet","shiny","shinyvalidate","s
hinyWidgets","shinythemes","tippy","shinyjs")
#install.packages(my_packages)
lapply(my_packages, require, character.only = T)
```
~~~

The code begins with setting the UI using fluidPage() function and applying the "lumen" theme to it. The useShinyjs() function is used to include the shinyjs package, which enables the use of JavaScript in the Shiny app. The code then defines a style for the background of the webpage using tags$style() function and sets the background image, size, position, and repeat properties.

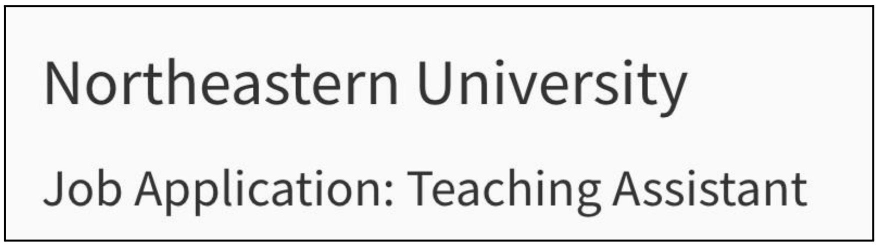


```
ui <- fluidPage(theme = shinytheme("lumen"),
    useShinyjs(),
    tags$style(
    HTML(
      "
      body {
          background-image:
url('https://images.credly.com/images/432ea12d-444b-42e7-a1d9-f5a3655fb948/blob.png');
          background-size: 45%;
          background-repeat: no-repeat;
          background-position: right;
      }
      "
    )
  ),
```

The sidebarLayout() function defines the layout of the app with a sidebar panel on the left and a main panel on the right. The sidebar panel contains a title panel with the name of the university and the position applied for, followed by candidate details, address, email, phone number, education, experience, availability, interview availability, candidate status, salary expectation, and resume upload fields.

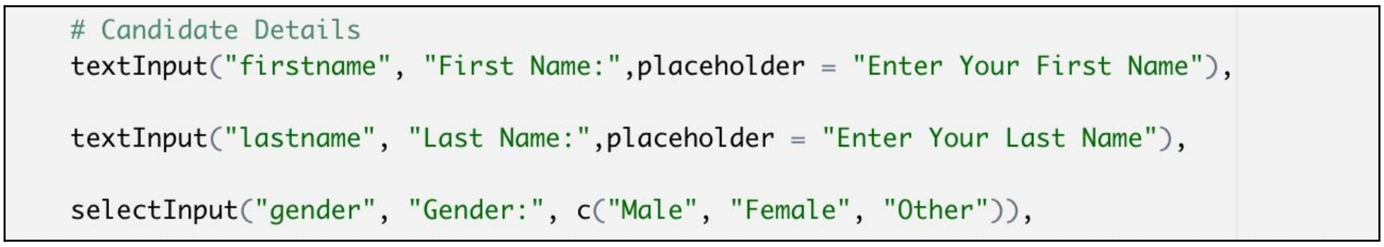


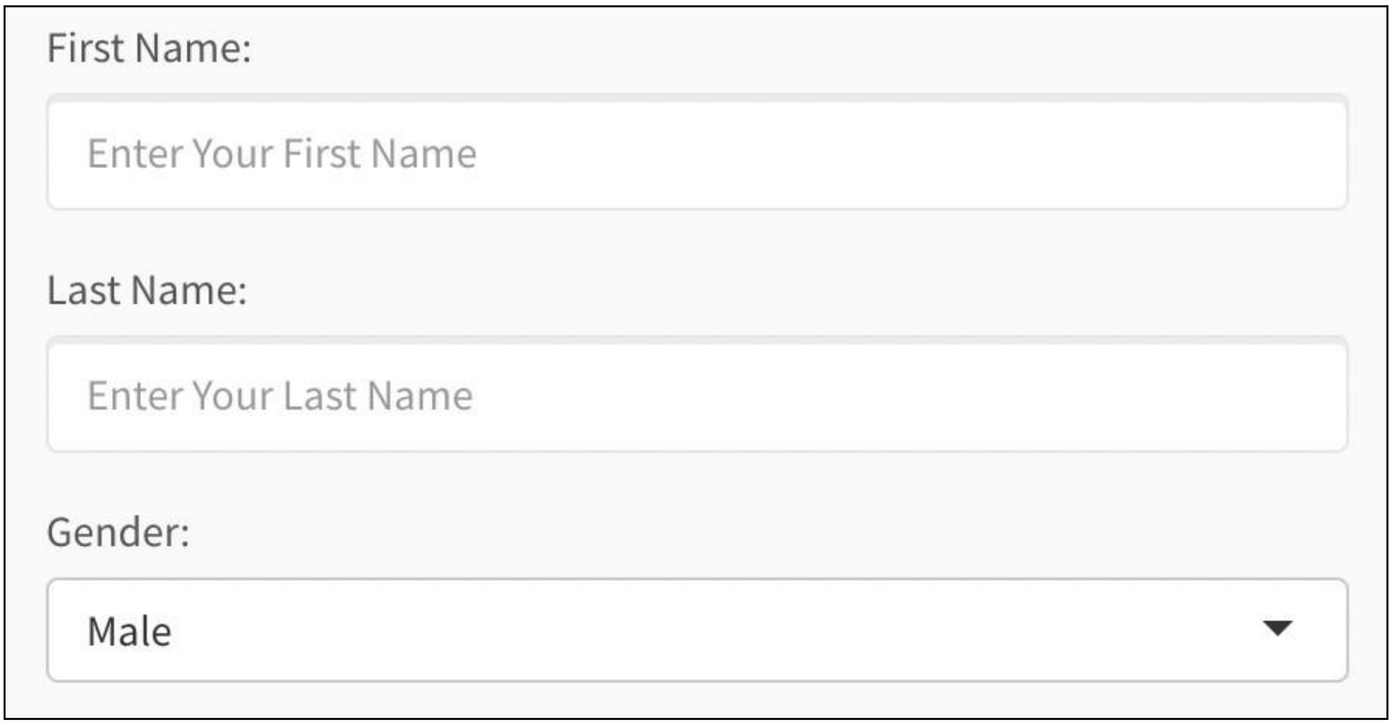
```
  sidebarLayout(
    sidebarPanel(
    titlePanel("Northeastern University"),
    h3("Job Application: Teaching Assistant"),
```


Northeastern University

Job Application: Teaching Assistant

The candidate details section consists of textInput() and selectInput() functions for the candidate's first name, last name, and gender. The Sections First Name and Last Name are Text Box and the Gender Section is a Top Down Option.


```
# Candidate Details
textInput("firstname", "First Name:",placeholder = "Enter Your First Name"),

textInput("lastname", "Last Name:",placeholder = "Enter Your Last Name"),

selectInput("gender", "Gender:", c("Male", "Female", "Other")),
```


First Name:

Enter Your First Name

Last Name:

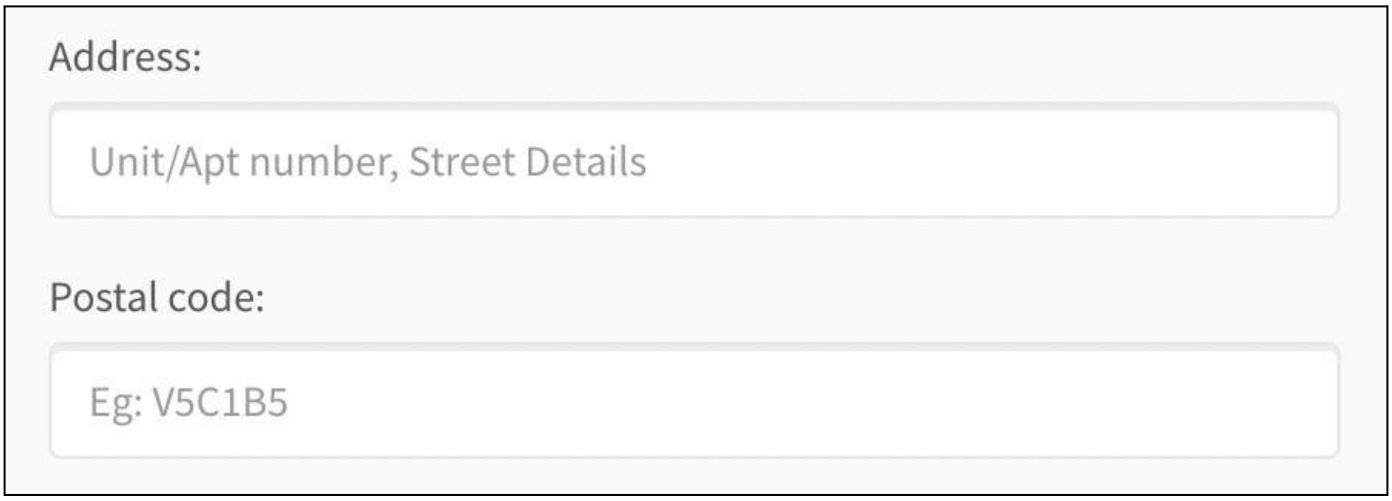
Enter Your Last Name

Gender:

Male

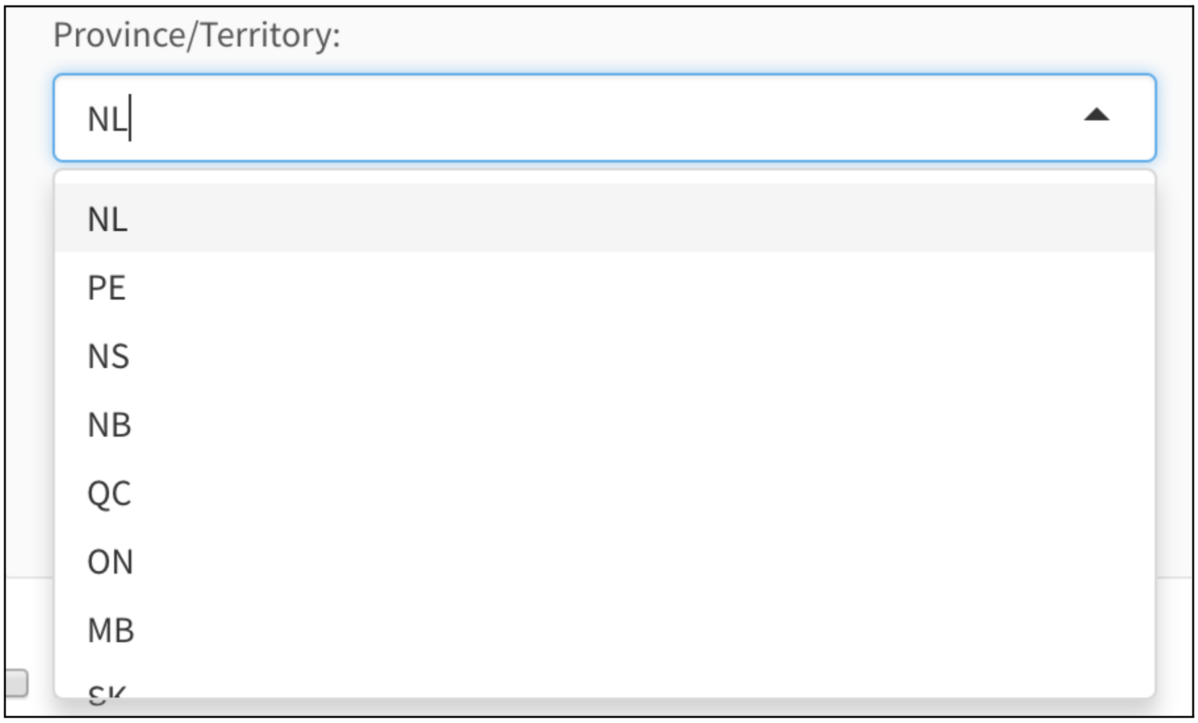
The address section has textInput() and tippy() functions for the candidate's address and postal code, respectively. The tippy() function adds a tooltip to the postal code input field. You can also see a Hover when you take the cursor on the Postal Code’s text box option. This makes it easier for the user to understand the format of the Postal Code.

```r
# Address
textInput("address", "Address:",placeholder = "Unit/Apt number, Street Details"),
tippy(
  textInput("postalcode", "Postal code:",placeholder = "Eg: V5C1B5"),
  tooltip = "Eg: V5C1B5"),
```



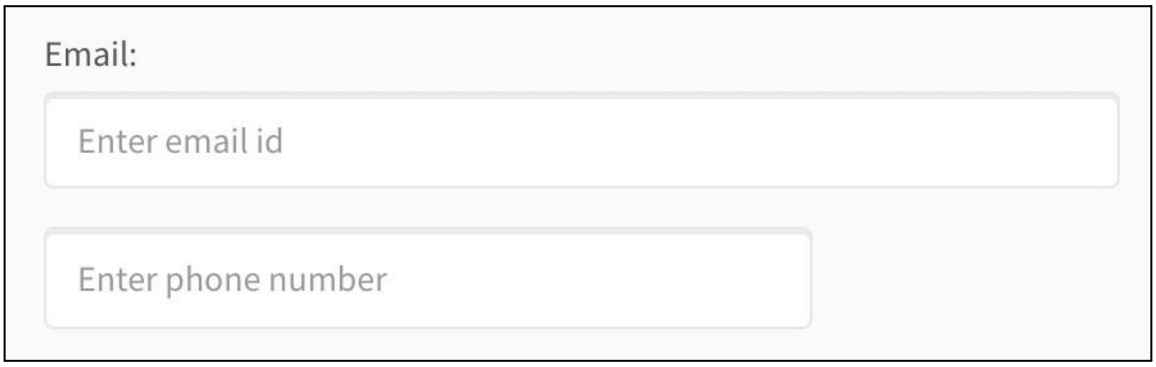
The provinces in Canada are listed in the selectInput() function. They give a drop-down menu using the shot form of the provinces which is universally accepted.

```r
  # Provinces in Canada
  selectInput(inputId = "province", "Province/Territory:",
              choices =
c("NL","PE","NS","NB","QC","ON","MB","SK","AB","BC","YT","NT","NU")),
```



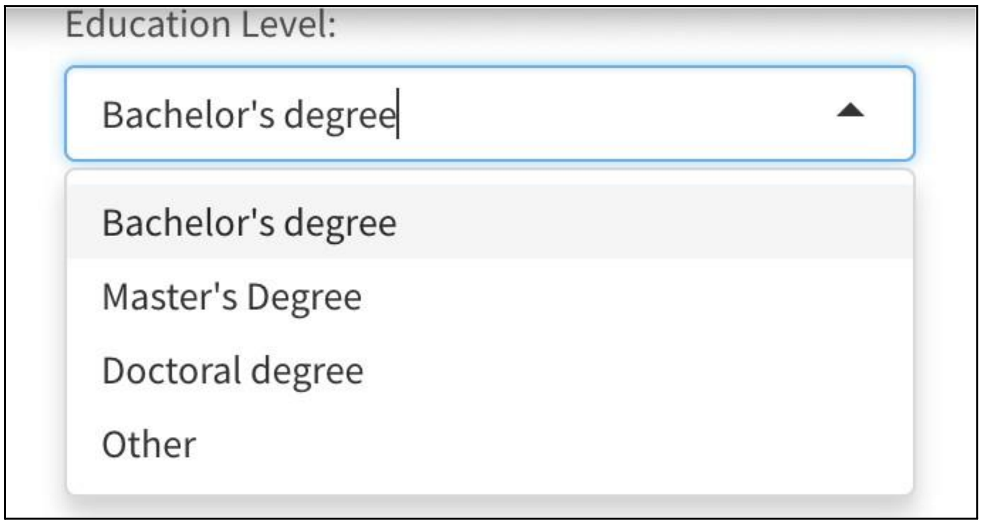
The email section has a tippy() function for the candidate's email address with a tooltip that reminds the user to enter a valid email ID. The phone number field uses div() and tags$input() functions to create an input field that restricts the user to enter only ten digits. The tags$script() function adds a placeholder to the phone number input field.

```r
    # Email id
    tippy(
      textInput("email", "Email:",placeholder = "Enter email id"),
      tooltip = "Enter a valid Email ID"),

    # Phone number
    div(
      id = "phone-wrapper",
      tags$input(
        id = "phone",
        type = "number",
        min = 0,
        max = 9999999999,
        step = 1,
        class = "form-control",
        style = "width: 300px; height: 40px;", # add CSS styling to adjust size
        oninput = "this.value = Math.max(0, Math.min(9999999999,
this.value)).toString().slice(0,10)"
      ),
      tags$script(
        "$('#phone').attr('placeholder', 'Enter phone number');"
      )
    ),

  ),
```



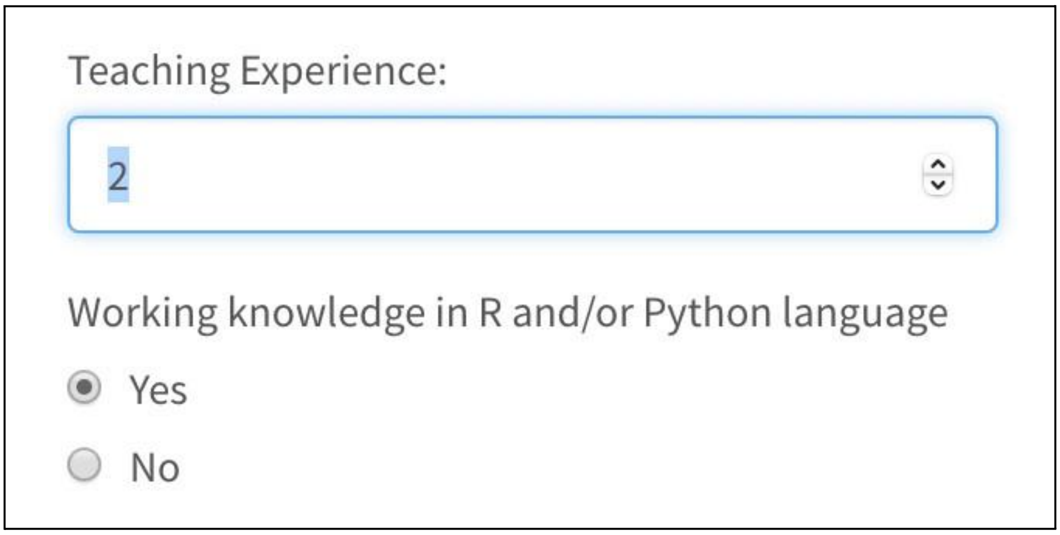
Next, in the Main Panel, we will create an “Education” option that has the options of ‘Bachelor’s’, ‘Master’s’, ‘Doctoral’ Degree or Other.

```r
    # Education
    selectInput(inputId = "edu", label = "Education Level:",
                choices = c("Bachelor's degree",
                            "Master's Degree",
                            "Doctoral degree",
                            "Other")),
```



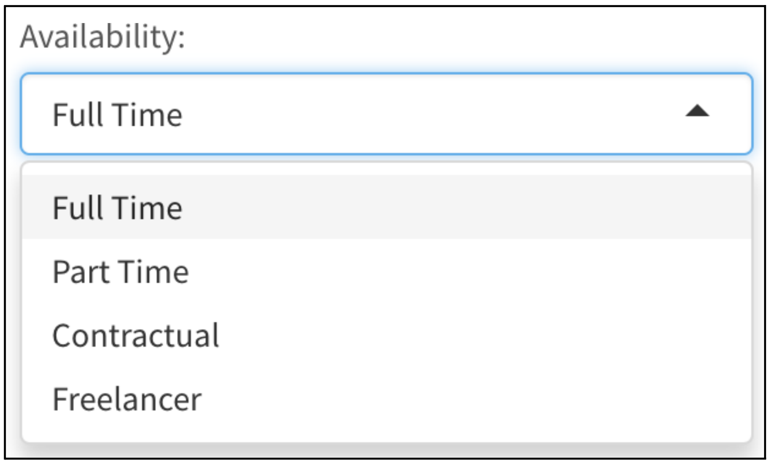
For Work Experience, we have kept the Criteria as a Minimum of Two Years and hence the Numeric Input starts from two instead of zero. We also have a Radio button asking if the person who is applying has worked in one of the languages, R or Python.

```
      # Experience
      verbatimTextOutput("value"),
      numericInput("Exp","Teaching Experience:", value = NULL,min=2),
      radioButtons("radio", "Working knowledge in R and/or Python language", choices =
c("Yes", "No")),
```



The Person applying can also select his/her availability as ‘Full Time’, ‘Part Time’, ‘Contractual’, or ‘Freelancer’. The selectInput() function allows a drop-down menu to choose the following options as follows:

```
    # Availability
    selectInput(inputId = "Availability", "Availability:",
                choices = c("Full Time",
                            "Part Time",
                            "Contractual",
                            "Freelancer")),
```

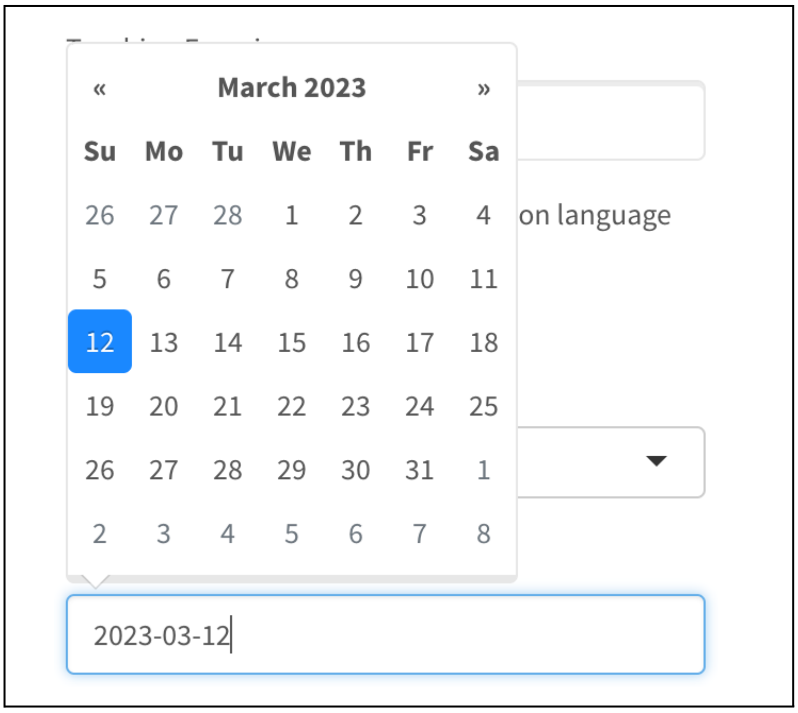

Availability:

Full Time

Full Time
Part Time
Contractual
Freelancer

The Date for Interview can also be selected using the dateInput() function. As you can see, it shows a calendar from which we can select the date accordingly.

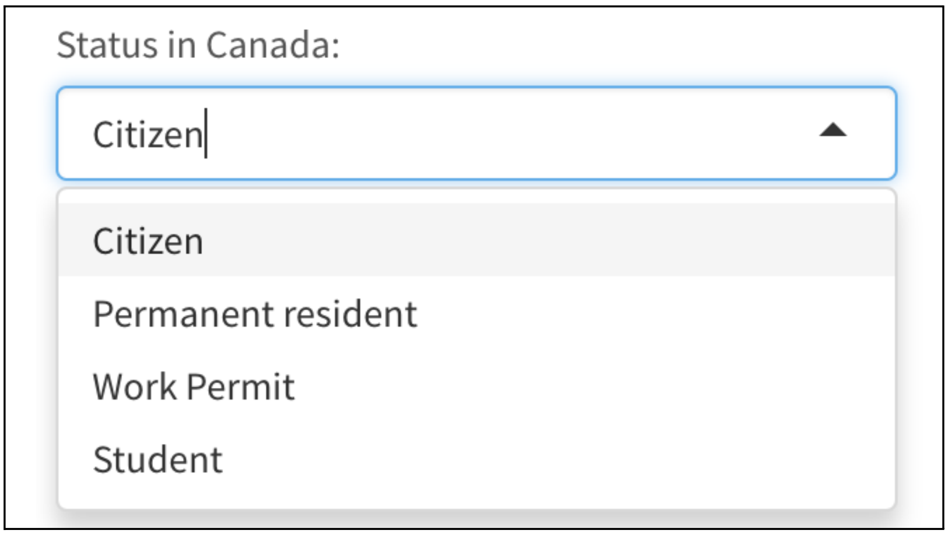
```
    # Interview Availability
    dateInput("date", "Choose Date For Interview:"),
```



2023-03-12

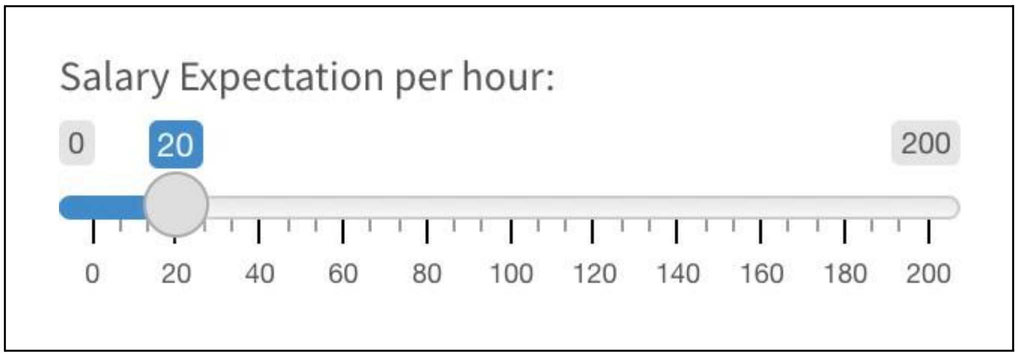
We have also asked for the Candidate Status and have given the following options. ‘Citizen’, ‘Permanent Resident’, ‘Work Permit’, and ‘Student’. The selectInput() function allows a drop-down menu to choose as follows:

```r
# Candidate Status
selectInput(inputId = "candidatestatus", "Status in Canada:",
            choices = c("Citizen",
                        "Permanent resident",
                        "Work Permit" ,
                        "Student")),
```

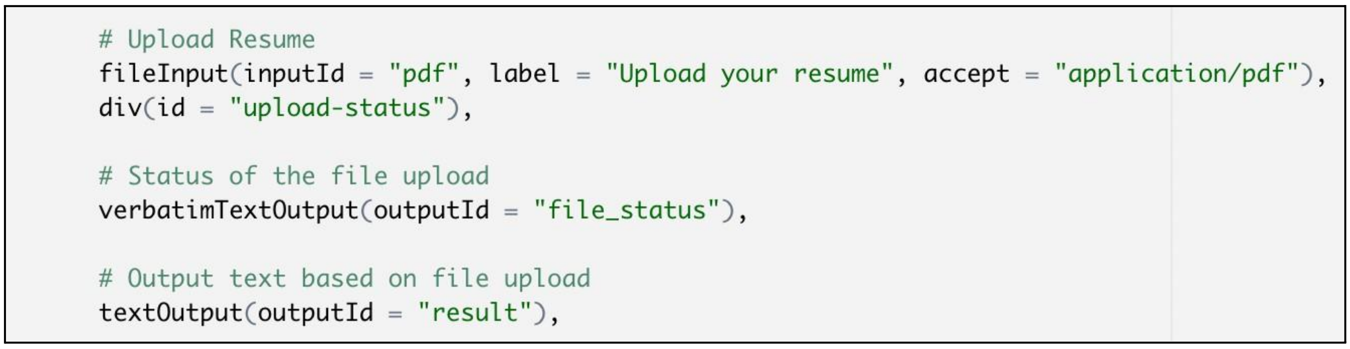


We have created a Salary Expectation part in our form using the sliderInput() function. The minimum and Maximum Values have been provided and the difference in scale is of 20.

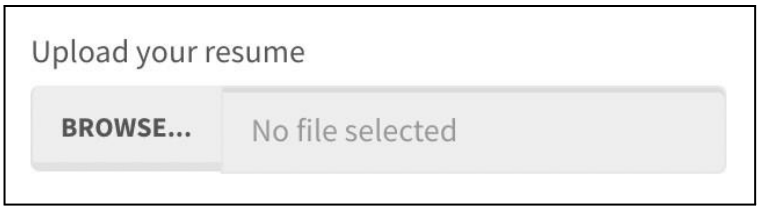
```r
#Salary
sliderInput("Salary", "Salary Expectation per hour:", min = 0, max = 200, value = 20),
```



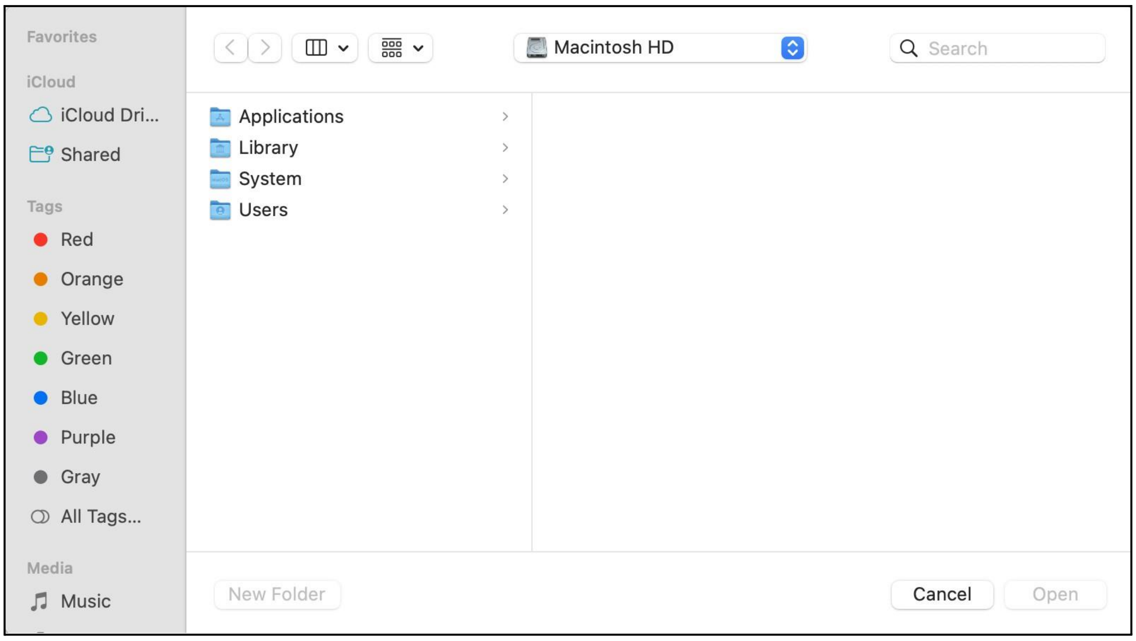
Finally, the person applying shall be asked to upload his/her resume. We have created an accept condition for .pdf files only. Thus, the resume can be uploaded in a PDF format as follows.



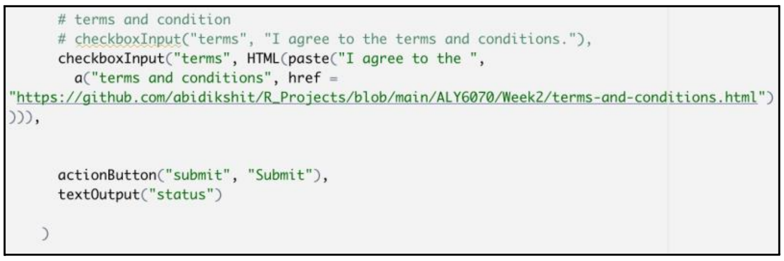
```
    # Upload Resume
    fileInput(inputId = "pdf", label = "Upload your resume", accept = "application/pdf"),
    div(id = "upload-status"),

    # Status of the file upload
    verbatimTextOutput(outputId = "file_status"),

    # Output text based on file upload
    textOutput(outputId = "result"),
```



Upload your resume



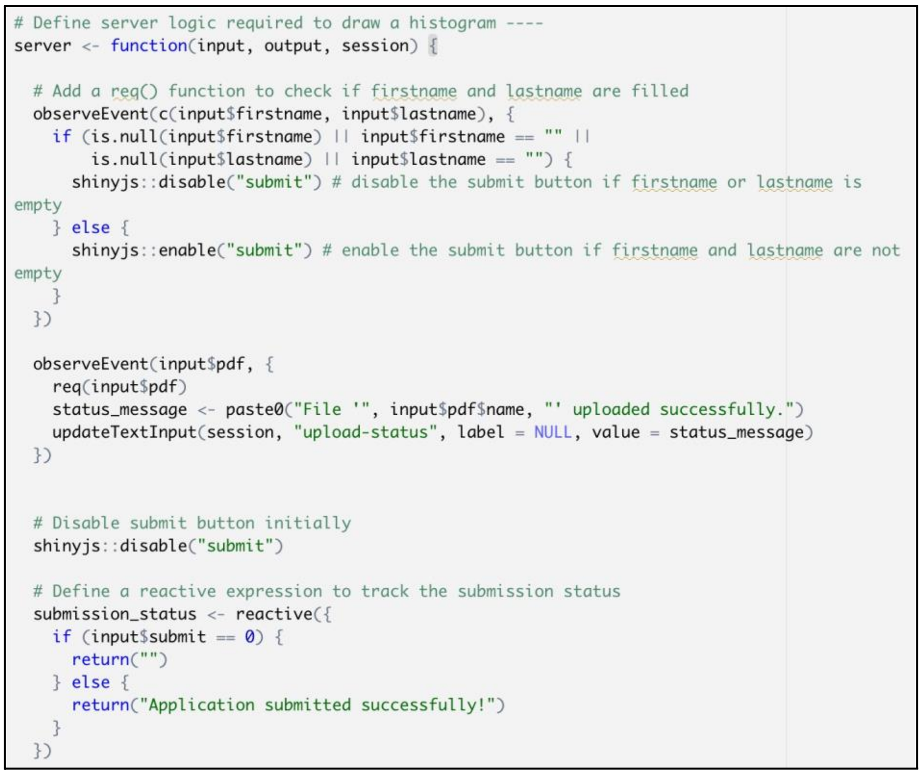
Finally, we shall ask the person to Agree to the Terms and Conditions and then submit the resume. To submit the resume, we have some criteria's that need to be filled. The Input areas firstname, last name, email, and files with .pdf extension are required for the Radio Button (Submit Button) to work.



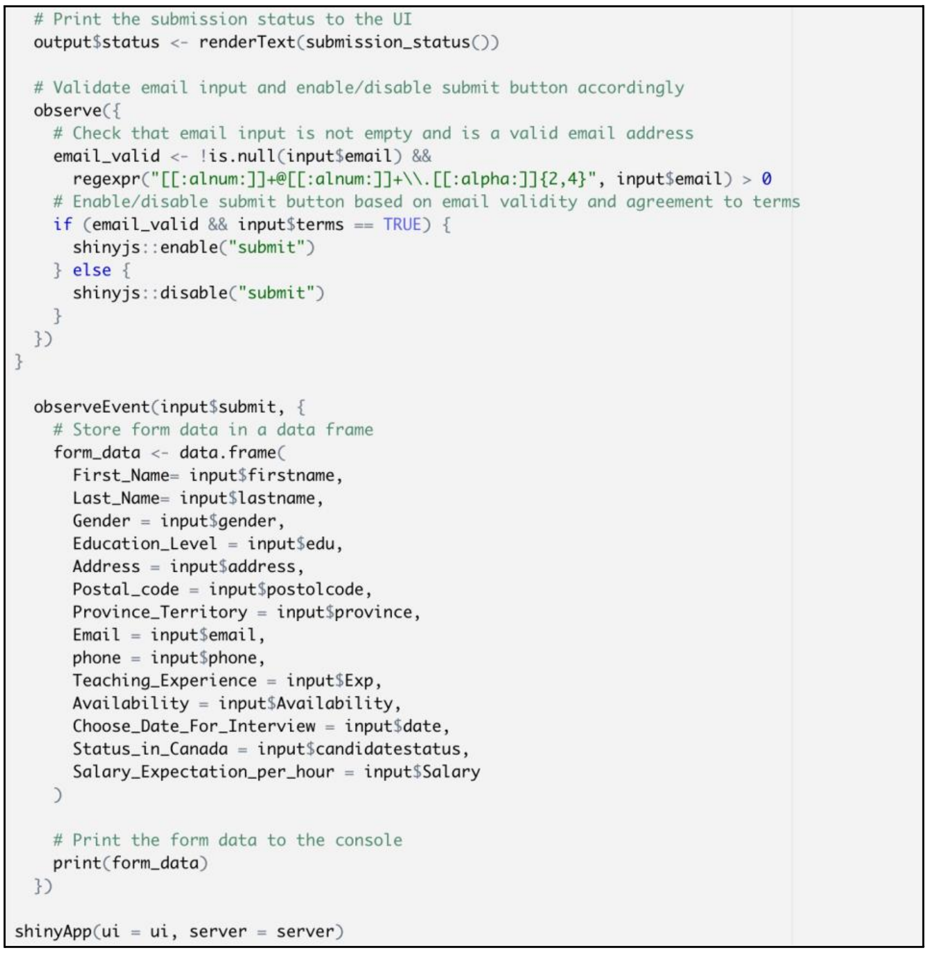
```
    # terms and condition
    # checkboxInput("terms", "I agree to the terms and conditions."),
    checkboxInput("terms", HTML(paste("I agree to the ",
      a("terms and conditions", href =
"https://github.com/abidikshit/R_Projects/blob/main/ALY6070/Week2/terms-and-conditions.html")
))),


    actionButton("submit", "Submit"),
    textOutput("status")

  )
```

I agree to the terms and conditions

Once we have created the UI part of the code, we shall create the Server part. The code for the following looks as follows:

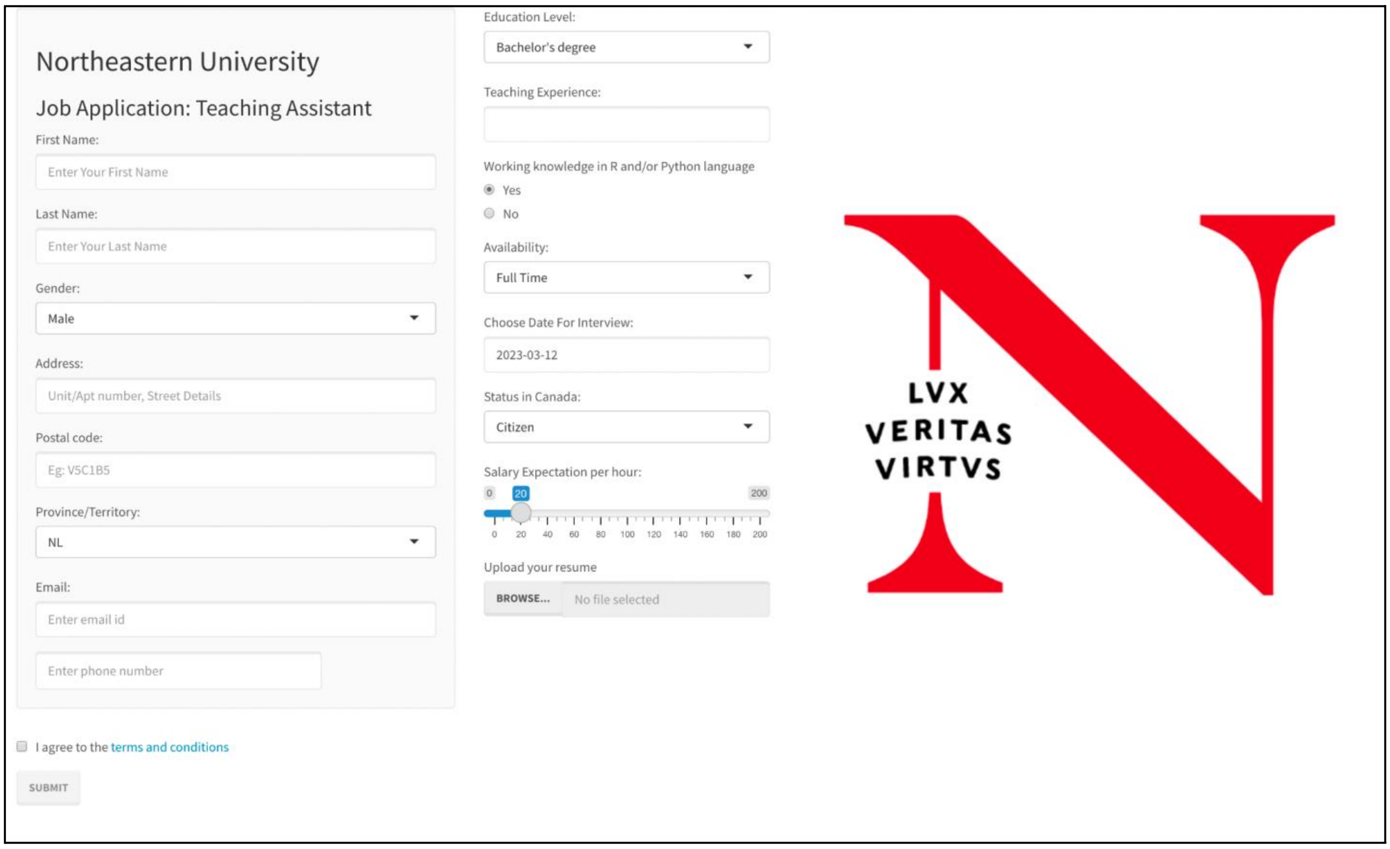


```r
# Define server logic required to draw a histogram ----
server <- function(input, output, session) {

  # Add a req() function to check if firstname and lastname are filled
  observeEvent(c(input$firstname, input$lastname), {
    if (is.null(input$firstname) || input$firstname == "" ||
        is.null(input$lastname) || input$lastname == "") {
      shinyjs::disable("submit") # disable the submit button if firstname or lastname is
empty
    } else {
      shinyjs::enable("submit") # enable the submit button if firstname and lastname are not
empty
    }
  })

  observeEvent(input$pdf, {
    req(input$pdf)
    status_message <- paste0("File '", input$pdf$name, "' uploaded successfully.")
    updateTextInput(session, "upload-status", label = NULL, value = status_message)
  })


  # Disable submit button initially
  shinyjs::disable("submit")

  # Define a reactive expression to track the submission status
  submission_status <- reactive({
    if (input$submit == 0) {
      return("")
    } else {
      return("Application submitted successfully!")
    }
  })
```



```r
  # Print the submission status to the UI
  output$status <- renderText(submission_status())

  # Validate email input and enable/disable submit button accordingly
  observe({
    # Check that email input is not empty and is a valid email address
    email_valid <- !is.null(input$email) &&
      regexpr("[[:alnum:]]+@[[:alnum:]]+\\.[[:alpha:]]{2,4}", input$email) > 0
    # Enable/disable submit button based on email validity and agreement to terms
    if (email_valid && input$terms == TRUE) {
      shinyjs::enable("submit")
    } else {
      shinyjs::disable("submit")
    }
  })
}

  observeEvent(input$submit, {
    # Store form data in a data frame
    form_data <- data.frame(
      First_Name= input$firstname,
      Last_Name= input$lastname,
      Gender = input$gender,
      Education_Level = input$edu,
      Address = input$address,
      Postal_code = input$postolcode,
      Province_Territory = input$province,
      Email = input$email,
      phone = input$phone,
      Teaching_Experience = input$Exp,
      Availability = input$Availability,
      Choose_Date_For_Interview = input$date,
      Status_in_Canada = input$candidatestatus,
      Salary_Expectation_per_hour = input$Salary
    )

    # Print the form data to the console
    print(form_data)
  })

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

The entire form after creating the UI and Server part together looks as follows:


Northeastern University

Job Application: Teaching Assistant

First Name: Enter Your First Name

Last Name: Enter Your Last Name

Gender: Male

Address: Unit/Apt number, Street Details

Postal code: Eg: V5C1B5

Province/Territory: NL

Email: Enter email id

Enter phone number

Education Level: Bachelor's degree

Teaching Experience:

Working knowledge in R and/or Python language
● Yes
○ No

Availability: Full Time

Choose Date For Interview: 2023-03-12

Status in Canada: Citizen

Salary Expectation per hour: 0 20 200

Upload your resume: BROWSE... No file selected

☐ I agree to the terms and conditions

SUBMIT

Bibliography

[1] Chang, W., Cheng, J., Allaire, J. J., Xie, Y., & McPherson, J. (2021). Shiny: Web Application Framework for R. R Foundation for Statistical Computing. Retrieved from https://CRAN.R-project.org/package=shiny

[2] *What is Shiny (in R)? | Domino Data Science Dictionary*. (n.d.). https://www.dominodatalab.com/data-science-dictionary/shiny-in-r

[3] Comprehensive R Archive Network (CRAN). (2021, October 21). *Data Insights Through Inbuilt R Shiny App [R package shinyr version 0.3.0]*. https://cran.r-project.org/web/packages/shinyr/index.html