MANIPAL UNIVERSITY
JAIPUR

School of Computing and Information Technology

Department of Computer Science and Engineering

B.TECH.– DESIGN ANALYSIS & ALGORITHM PROJECT REPORT

BY:

SOUMIK MALLIK - 169105194
SAHAL TANUJ SANDEEP - 169105205

1. Title of the Project:

UI based Prim's algorithm

2. Statement about the Problem:

Create a portal [GUI] in which user can upload excel files or a CSV file which contains the data of the connected graph in a manner [vertex1, vertex2, edge weight]. The portal should be able to display the original graph long with the minimum spanning tree graph using Prim's algorithm.

3. Why is this particular topic chosen?

This topic is chosen because Prim's algorithm is significantly faster in the limit when you've got a really dense graph with many more edges than vertices. So working on this project helped us in understanding the Prim algorithm in a better way by making its user interface and along with that helped us in learning R which is a Data science language in this project to make a user interface so that project can be scaled in the future.

4. Objective and scope of the project:

Objective:-

Our objective is to find the minimum cost spanning tree using a Prim's Algorithm.

Scope:-

Our project can be used to study for ways to lay out electrical networks in a way that minimizes the total cost of the wiring. In a minimum spanning tree, all the nodes(houses) would be connected to power by wires in a way that has minimum cost and redundancy (cutting any wire necessarily cuts the power grid into two pieces.)

It can be also used to generate mazes. Prim's algorithm has been used this way for creating high-quality mazes.

5. Methodology

Source Code:-

ui.R

```
1. library(shiny)
2. shinyUI(fluidPage(
3.   titlePanel(title = h2("Demonstration of Prim Algorithm", align = "center")),
4.   sidebarLayout(

            sidebarPanel(

5.          fileInput("file","Upload the file"), # fileinput() function is used to get the file upload
    contorl option
6.          helpText("Default max. file size is 5MB"),
7.          tags$hr(),
8.          h5(helpText("Select the read.table parameters below")),
9.          checkboxInput(inputId = 'header', label = 'Header', value = FALSE),
10.         checkboxInput(inputId = "stringAsFactors", "stringAsFactors", FALSE),
11.         br(),
12.         radioButtons(inputId = 'sep', label = 'Separator', choices =
    c(Comma=',',Semicolon=';',Tab='\t', Space=''), selected = ',')

            ),
```

```
        mainPanel(

13.     uiOutput("tb")
14.
15.




        )

16.
17.  )
18. ))
```

In ui.R, we have made the Frontend of the project, where the user uploads a CSV file of the format (weight1, weight2, vertex). In the UI , there are three main parts,

1) **Title Panel:- The Header of the project**
2) **Sidebar Layout:- Where the user can upload the CSV file and can select options for displaying the data file.**
3) **Main Layout:- where the main MST graph is displayed.**
    a) **The red coloured edges are part of MST.**
    b) **The black colour edges are part of main graph.**

Server.R

```
1. library(shiny)
2. library(rio)
3. library(optrees)
4. # use the below options code if you wish to increase the file input limit, in this example
   file input limit is increased from 5MB to 9MB
5. # options(shiny.maxRequestSize = 9*1024^2)
6.
7. shinyServer(function(input,output){
8.
9.   # This reactive function will take the inputs from UI.R and use them for read.table() to
   read the data from the file. It returns the dataset in the form of a dataframe.
10.  # file$datapath -> gives the path of the file
11.  data <- reactive({
        a. file1 <- input$file
        b. if(is.null(file1)){return()}
```

```
        c. read.table(file=file1$datapath, sep=input$sep, header = input$header,
           stringsAsFactors = input$stringAsFactors)
12.
13. })
14.
15. # this reactive output contains the summary of the dataset and display the summary in
    table format
16. output$filedf <- renderTable({
17. if(is.null(data())){return ()}
18.  input$file
19. })
20.
21. # this reactive output contains the summary of the dataset and display the summary in
    table format
22. output$sum <- renderTable({
        a. if(is.null(data())){return ()}
        b. summary(data())
23.
24. })
25.
26. # This reactive output contains the dataset and display the dataset in table format
27. output$table <- renderTable({
        a. if(is.null(data())){return ()}
        b. data()
28. })
29.
30. output$PrimGraph <- renderPlot({
        a. y<-data()
        b. a<-max(max(y$v1),max(y$v2))
        c. y<- as.matrix(y)
        d. x<- 1:a
        e. getMinimumSpanningTree(x, y,"Prim", start.node = 1, show.data = TRUE,
           show.graph = TRUE, check.graph = FALSE)
31.
32. })
33.
34. output$PrimTable <- renderPrint({
        a. y<-data()
        b. a<-max(max(y$v1),max(y$v2))
        c. y<- as.matrix(y)
        d. x<- 1:a
        e. getMinimumSpanningTree(x, y,"Prim", start.node = 1, show.data = TRUE,
           show.graph = FALSE, check.graph = FALSE)
```

```
35.
36.  })
37.
38.  # the following renderUI is used to dynamically generate the tabsets when the file is
     loaded. Until the file is loaded, app will not show the tabset.
39.  output$tb <- renderUI({
        a. if(is.null(data()))
40.          h5("Tanuj Sahal and Soumik Mallik")
        a. else
41.          tabsetPanel(tabPanel("About file", tableOutput("filedf")),tabPanel("Data",
     tableOutput("table")),tabPanel("Summary", tableOutput("sum")),tabPanel("Generate
     MST Graph", plotOutput("PrimGraph")),tabPanel("Show MST Table",
     textOutput("PrimTable")))
42.  })
43. })
```

The server.R is the backend of our project. In the backend, we have prepared functions for each tab or button. Like when the user uploads the CSV file and clicks on following buttons and their corresponding functions are called:

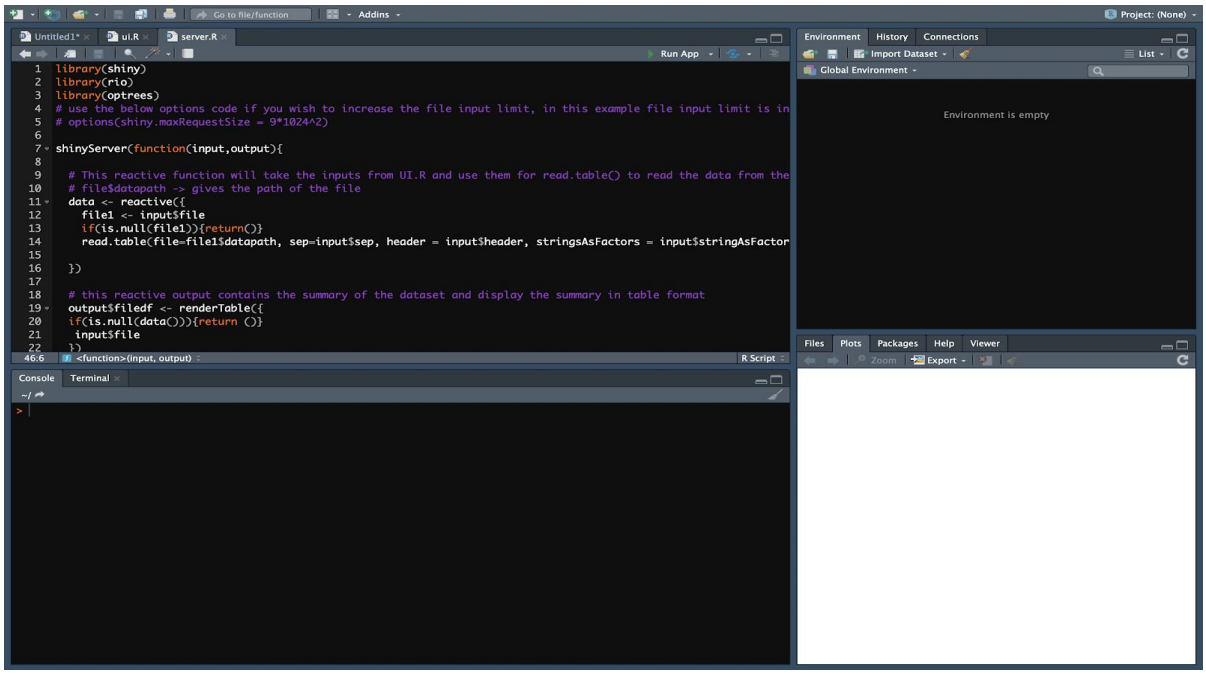
1. **About file : output$filedf** : run the code for displaying the name, size, type and datapath of the data file.
2. **Data : output$table :** run the code for displaying the data of .csv file uploaded.
3. **Summary : output$sum :** run the code for displaying the summary of each coloumn of data like max, mean, median values.
4. **Generate MST graph : output$PrimGraph :** run the code for displaying the MST graph in which red coloured edges are part of MST and other edges are part of main graph.
5. **Show MST table : output$PrimTable :** run the code for displaying the matrix data of the MST graph.

6. Hardware & Software to be used

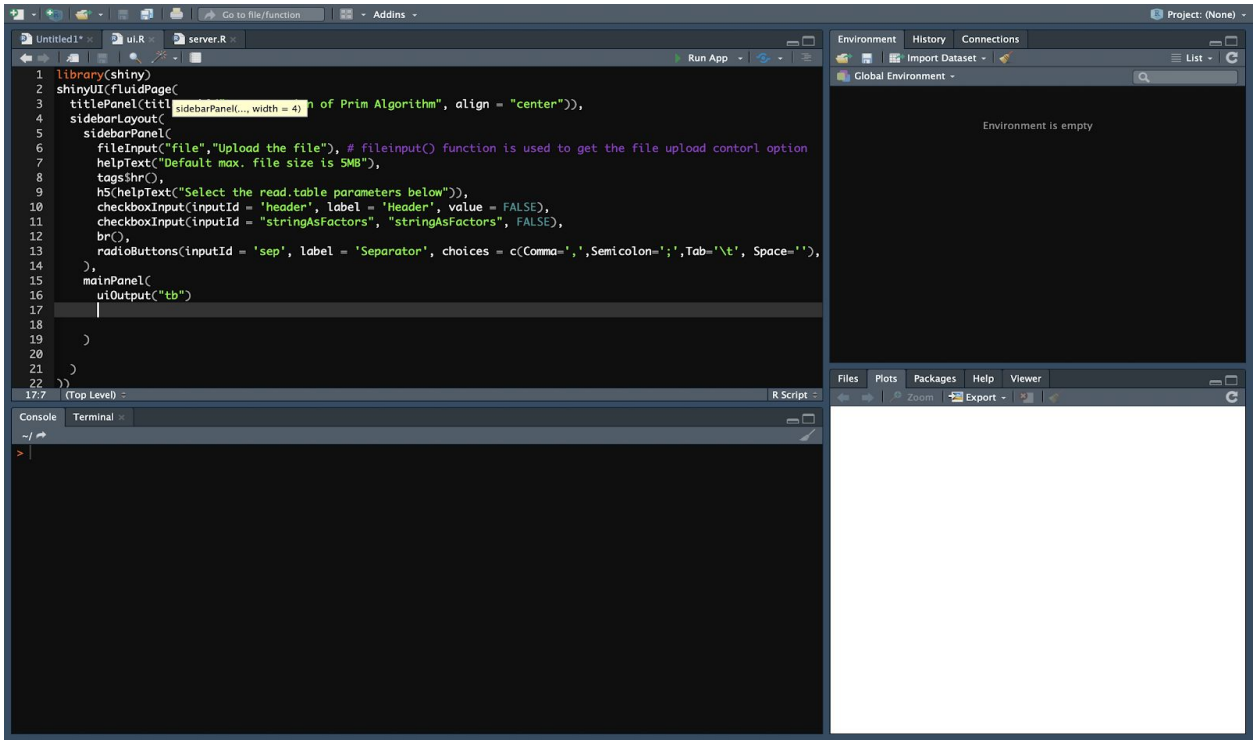
Hardware:- Laptop HP probook G3, Apple Macbook Pro

Software:- RStudio and RShiny

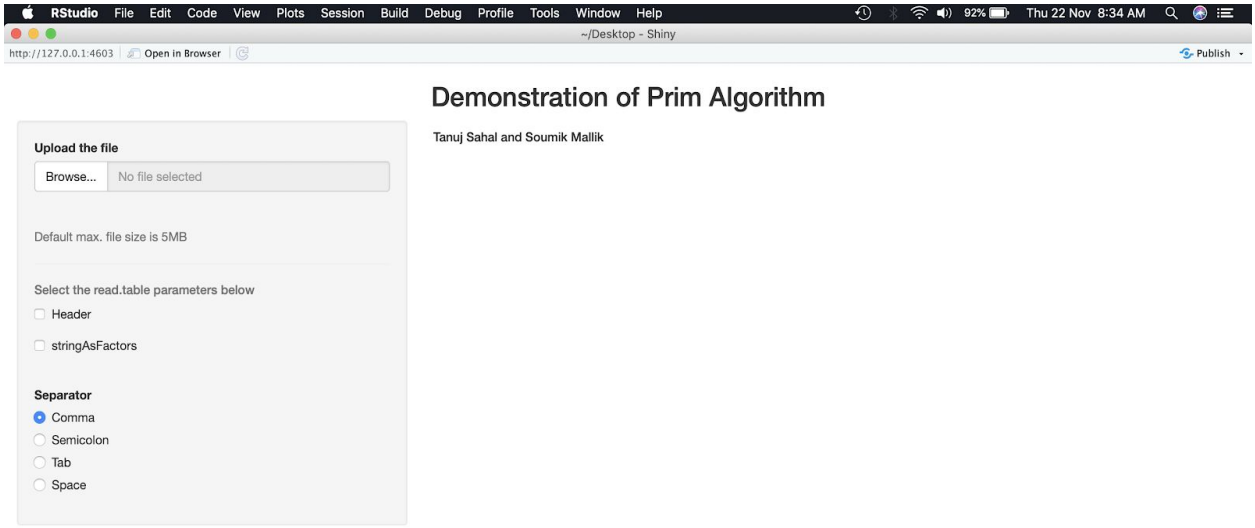
7. Screen Shots



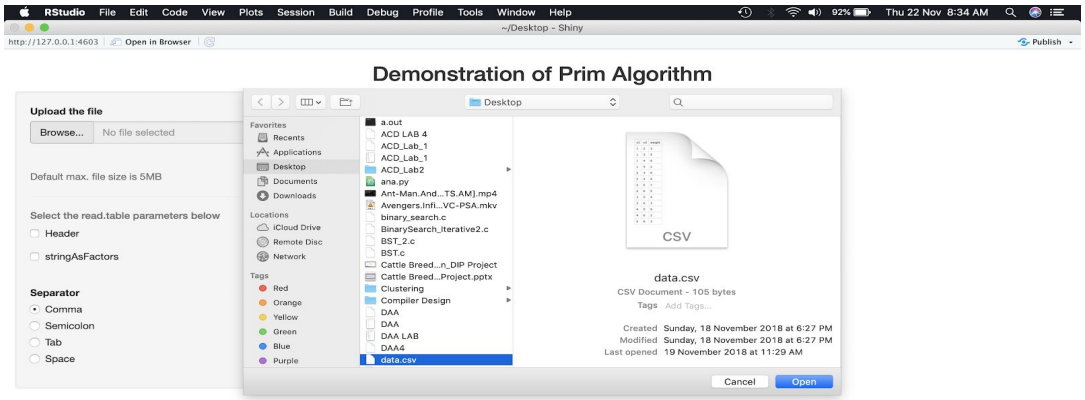
server.R (Backend)



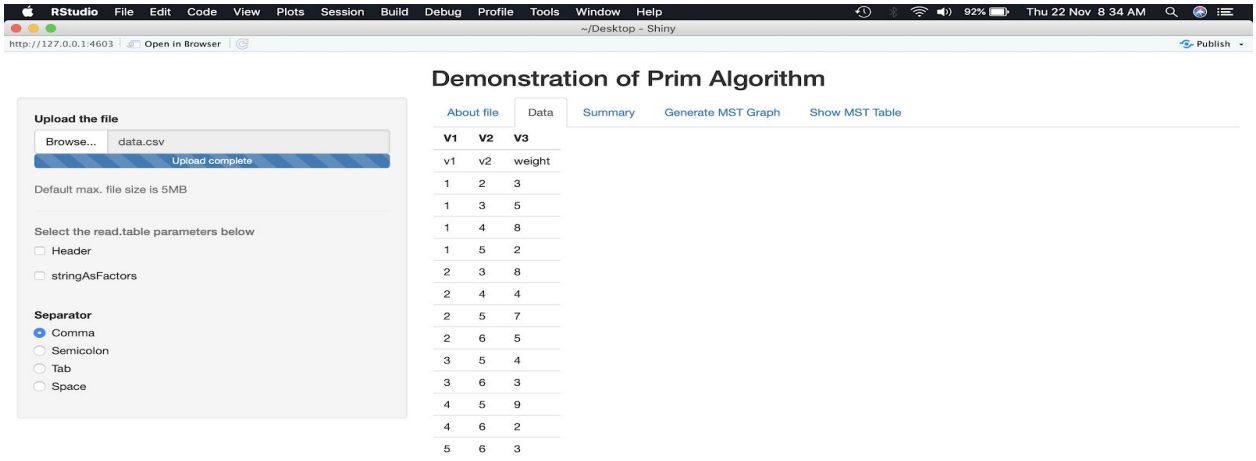
ui.R(Frontend)



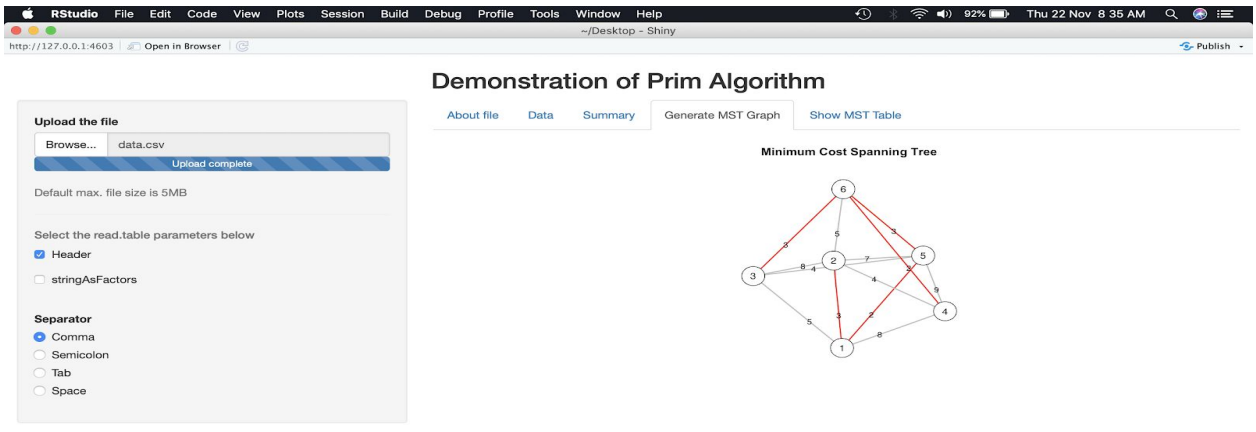
First Page of the Web UI



Uploading CSV File



After uploading csv file, Data is displayed



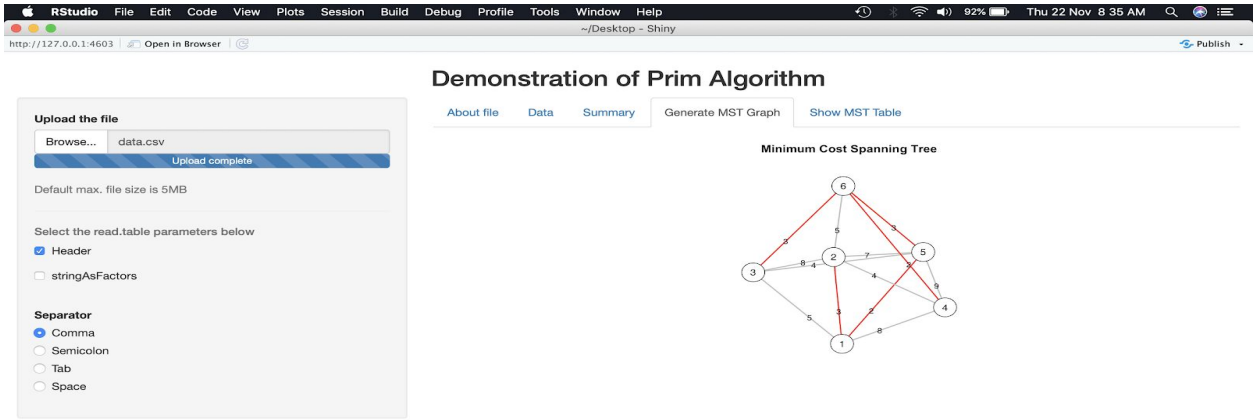
Final Output of the Graph

8. Test Cases:-

Input #1:-

v 1	v 2	weig ht
1	2	3
1	3	5
1	4	8
1	5	2
2	3	8
2	4	4
2	5	7
2	6	5
3	5	4
3	6	3
4	5	9
4	6	2
5	6	3

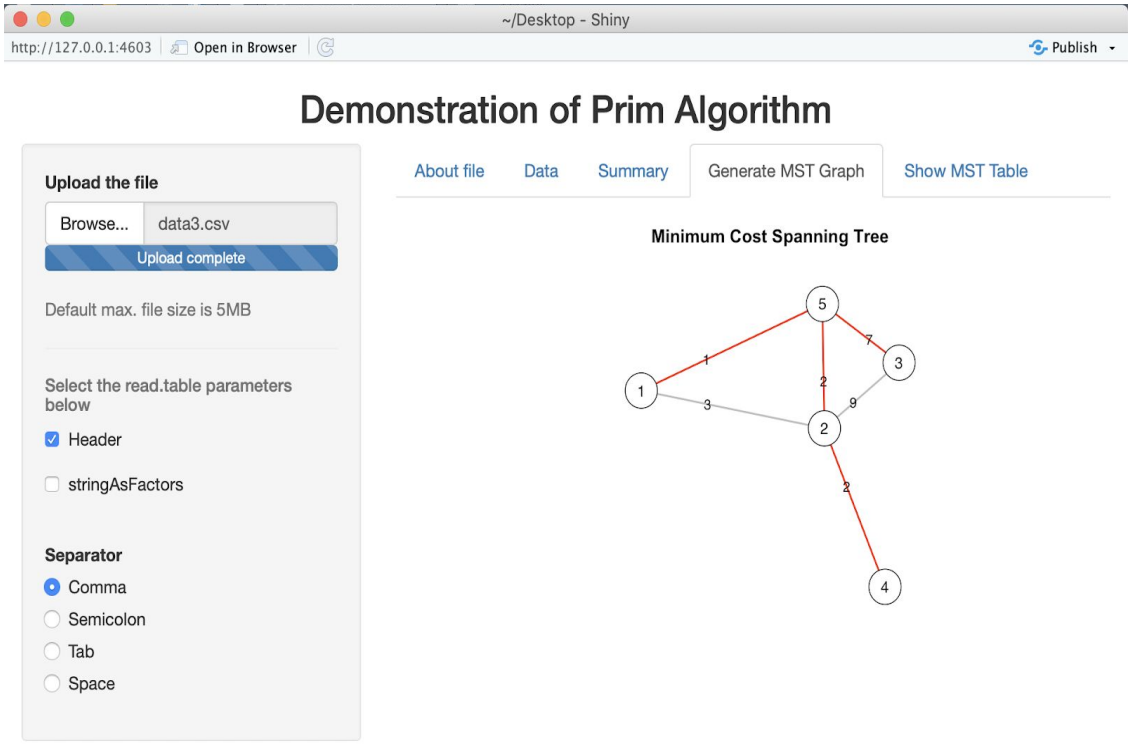
Output #1 :-



Input #2:-

v 1	v 2	weig ht
1	5	1
1	2	3
5	2	2
5	3	7
2	3	9
2	4	2

Output #2:-



8) What contribution would the project make?

Meeting the needs of rural people and thinking through what could potentially affect those who work and live in rural areas is vital for local authorities. The minimum spanning tree problem has important applications in network design which has been extensively studied in the literature. The minimum spanning tree problem on a graph with edge costs and vertex profits asks for a subtree maximizing the difference between the total cost of all edges in the subtree and the total profits of all vertices contained in the subtree. Minimum spanning tree problem appears in the design of utility networks (e.g. bus services, electrifications) where villages and the network connecting them have to be chosen in the most profitable way. This project demonstrates the application of Prim's algorithms to the design of local access networks. Our main contribution is an efficient algorithmic implementation using an undirected graph model.