#server.R Code

```r
#
# This is the server logic of a Shiny web application. You can run the
# application by clicking 'Run App' above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#

library(shiny)
library(tidyverse)
library(maps)
library(scales)
library(sf)
library(shinycssloaders)
lendingClubLoanData <- read.csv("data/lending_club_loan_data_final.csv")
fullStateNames <- read.csv("data/states.csv")
states <- st_as_sf(map("state", plot = FALSE, fill = TRUE))
incomeLabels <- c('0-20K','20-40K','40-60K','60-80K','80-100K','100-120K',
                  '120-140K','140-160K','160-180K','180-200K', '200-220K',
                  '220-240K', '240-260K', '260-280K', '280-300K')


loan_statuses <- c("Current",
                   "Fully Paid",
                   "Late (31-120 days)",
                   "In Grace Period",
                   "Charged Off",
                   "Late (16-30 days)")

filteredLendingClubData <- lendingClubLoanData %>%
  drop_na(annual_inc)

filtereDdtiData <- lendingClubLoanData %>%
  drop_na(dti)%>%
  drop_na(dti) %>%
  filter(dti < 100)

filteredAnnualIncomeData <- lendingClubLoanData %>%
  drop_na(annual_inc)%>%
  filter(annual_inc < 300000)


# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  loanStatusFilter <- function(loanStatusValue) {
    toReturn <- c("Current",
                  "Fully Paid",
                  "Late (31-120 days)",
                  "In Grace Period",
                  "Charged Off",
                  "Late (16-30 days)",
```

```r
                "Default",
                "Does not meet the credit policy. Status:Fully Paid",
                "Does not meet the credit policy. Status:Charged Off")
  if (loanStatusValue != "Any"){
    toReturn <- c(loanStatusValue)
  }
  return(toReturn)
}

number_of_loans_each_year <- reactive({
  numberOfLoansEachYear <- lendingClubLoanData %>%
    filter (loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%
    group_by(orig_year) %>%
    summarise(loanCountByYear=n())
})

total_amount_funded_each_year <- reactive({
  totalFundedAMountPerYear <- lendingClubLoanData %>%
    filter (loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%
    group_by(orig_year)%>%
    summarise(totalFundedAmount= sum(as.numeric(funded_amnt)))
})

loan_amt_term_relation <- reactive({
  lendingClubLoanData %>%
    filter (loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus))
})

dti_trend <- reactive({
  filtereDdtiData %>%
    filter(loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus))
})

funded_amt_term_interest_relation <- reactive({
  filteredLendingClubData <- filteredLendingClubData %>%
    filter(annual_inc <= 300000) %>%
    filter(loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus))
```

```r
  groupedData <- filteredLendingClubData %>%
    group_by(incomeGroup = cut(annual_inc, breaks= seq(0, 300000, by = 20000),
                               right = TRUE, include.lowest = TRUE, labels = incomeLabels) ) %>%
    summarise(averageInterest= mean(int_rate), averageLoanLoanFundedAmount = mean(funded_amnt))

})

income_trend <- reactive({
  filteredAnnualIncomeData %>%
    filter (loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus))
})

loan_funded_amt_by_state <- reactive({
  fundedAmountByState <- lendingClubLoanData %>%
    filter (loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%
    group_by(addr_state)%>%
    summarise(totalFundedAmount= sum(as.numeric(funded_amnt)))

  fundedAmountByState <- fundedAmountByState %>%
    inner_join(fullStateNames, by = c("addr_state" = "abbreviation"))

  states2 <- states %>% left_join(fundedAmountByState, by = c("ID" = "state" ))
})

loans_by_status <- reactive({
  numberOfLoansByLoanStatus <- lendingClubLoanData %>%
    filter(loan_status %in% loan_statuses) %>%
    filter(loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%
    group_by(loan_status)%>%
    summarise(numberOfLoans = n())
})

loans_by_purpose <- reactive({
  numberOfLoansByPurpose <- lendingClubLoanData %>%
    filter(loan_amnt >= input$loanAmountRange[1] & loan_amnt <= input$loanAmountRange[2]) %>%
    filter(grade %in% input$grades) %>%
    filter(home_ownership %in% input$homeOwnerships) %>%
    filter(loan_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%
    group_by(purpose)%>%
    summarise(numberOfLoans = n())
})
```

```r
output$loanProcessesedEachYear <- renderPlot({
  ggplot(data = number_of_loans_each_year())+
    geom_line(color="steelblue", size=1.2, aes(x=orig_year,y=loanCountByYear))+
    geom_point(color="steelblue", size=2.5, aes(x=orig_year,y=loanCountByYear))+
    scale_x_continuous(breaks = number_of_loans_each_year()$orig_year) +
    scale_y_continuous(labels = scales::comma_format()) +
    labs(x="Year",y="# Number of loans",title="Loans processed year on year (2007-2018)")+
    theme_minimal()
})

output$totalFundedLoanAmountEachYear <- renderPlot({
  ggplot(data = total_amount_funded_each_year(), aes(x=orig_year, y=totalFundedAmount)) +
    geom_bar(stat="identity", width=0.5, fill = "steelblue") +
    scale_x_continuous(breaks = total_amount_funded_each_year()$orig_year) +
    scale_y_continuous(labels = scales::dollar) +
    labs(x = "Year", y = "Total funded loan amount",title="Total loan funded amount year on year (200
    theme_minimal()
})

output$loanAmtTermRelation <- renderPlot({
  ggplot(data = loan_amt_term_relation()) +
    geom_boxplot(aes(x=term, y=funded_amnt, color=term)) +
    scale_y_continuous(labels = scales::dollar) +
    labs(x = "Term", y = "Loan funded amount", title="Loan funded amount and term relation") +
    theme_minimal()
})

output$fundedAmtIncomeAndInterestRelation <- renderPlot({
  ggplot(data =funded_amt_term_interest_relation(), aes(x=incomeGroup, y=averageLoanLoanFundedAmount)
    geom_point(colour="steelblue", shape=16, aes(size=averageInterest)) +
    geom_smooth(aes(incomeGroup, averageLoanLoanFundedAmount, group = 1), method = "lm") +
    scale_y_continuous(labels = scales::dollar) +
    labs(x="Annual Income ($)",y="Average loan funded amount",
         title="Relation between funded loan Amount, income and interest rate")+
    guides(size=guide_legend("Average \ninterest rate (%)")) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle =50, hjust=0.75))+
    theme(legend.background = element_rect())
})

output$incomeTrend <- renderPlot({
  ggplot(data = income_trend(), aes(x = annual_inc)) +
    geom_density(fill="steelblue", color="steelblue", alpha=0.8) +
    geom_vline(aes(xintercept=median(annual_inc)),color="green", linetype="dashed", size=1) +
    scale_x_continuous(labels = scales::dollar) +
    scale_y_continuous(labels = function(x) format(x, scientific = FALSE)) +
    labs(x="Annual income",y="Density of loans",title="Annual income distribution") +
    theme_minimal()
})

output$loanFundedAmtByState <- renderPlot({
  ggplot(data = loan_funded_amt_by_state()) +
    geom_sf(aes(fill = totalFundedAmount)) +
```

```r
      scale_fill_viridis_c("Loan funded amount", labels = scales::dollar) +
      labs(title = "Total loan funded amount by state") +
      theme_minimal()
  })

  output$dtiTrend <- renderPlot({
    ggplot(data = dti_trend(), aes(x = dti)) +
      geom_density(fill="steelblue", color="steelblue", alpha=0.8) +
      geom_vline(aes(xintercept=median(dti)),color="green", linetype="dashed", size=1) +
      labs(x="Debt to Income Ratio (DTI) %",y="Density of loans",
           title="Loan distribution across DTI (Excluded > 100") +
      theme_minimal()
  })

  output$loansByStatus <- renderPlot({
    ggplot(loans_by_status(), aes(x=loan_status, y=numberOfLoans)) +
      geom_bar(stat="identity", width=0.5, fill = "steelblue") +
      geom_text(aes(label=numberOfLoans), vjust=-0.3, size=3.5) +
      scale_y_continuous(labels = scales::comma_format()) +
      labs(x = "Loan Status", y = "Number of Loans (#)",title="# Loans by status") +
      theme_minimal() +
      theme(axis.text.x = element_text(angle =50, hjust=0.75))
  })

  output$loansByPurpose <- renderPlot({
    ggplot(loans_by_purpose(), aes(x=purpose, y=numberOfLoans)) +
      geom_bar(stat="identity", width=0.5, fill = "steelblue") +
      scale_y_continuous(labels = scales::comma_format()) +
      labs(x = "Loan Purpose", y = "Number of Loans (#)",title="# Loans by purpose") +
      theme_minimal() +
      theme(axis.text.x = element_text(angle =50, hjust=0.75))
  })

})
```

#ui.R code

```r
#
# This is the user-interface definition of a Shiny web application. You can
# run the application by clicking 'Run App' above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#

library(shiny)
library(shinythemes)
library(shinycssloaders)
# Define UI for application that draws a histogram
shinyUI(fluidPage(
    theme = shinytheme("slate"),
    # Application title
```

```r
    titlePanel(""),
    titlePanel(title=div(style="display:inline-block;width:100%;",
                      img(src="homelogo.png", style="height:100px;"), "Data analysis (2007-2018)"),
            windowTitle = "Lending club data analysis"),

    # Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            # Price Range filter
            sliderInput("loanAmountRange",
                        "Loan Amount:",
                        pre = "$",
                        min = 500,
                        max = 40000,
                        value = c(500, 40000)),
            # Horizontal line ----
            tags$hr(),

            # House condition filter
            checkboxGroupInput(inputId = "grades",
                            label ="Grade:",
                            choiceNames = c("A", "B", "C", "D", "E", "F", "G"),
                            choiceValues = c("A", "B", "C", "D", "E", "F", "G"),
                            selected = c("A", "B", "C", "D", "E", "F", "G"),
                            inline = TRUE),
            # Horizontal line ----
            tags$hr(),

            # House condition filter
            checkboxGroupInput(inputId = "homeOwnerships",
                            label ="Home Ownership:",
                            choiceNames = c("ANY", "RENT", "MORTGAGE", "OWN"),
                            choiceValues = c("ANY", "RENT", "MORTGAGE", "OWN"),
                            selected = c("ANY", "RENT", "MORTGAGE", "OWN"),
                            inline = TRUE),
            # Horizontal line ----
            tags$hr(),


            # Number of bedrooms filter
            selectInput("loanStatus", "Loan Status:",
                        c(
                            "Any" = "Any",
                            "Current" = "Current",
                            "Fully Paid" = "Fully Paid",
                            "Late (31-120 days)" = "Late (31-120 days)",
                            "In Grace Period" = "In Grace Period",
                            "Charged Off" = "Charged Off",
                            "Late (16-30 days)" = "Late (16-30 days)",
                            "Default" = "Default",
                            "Does not meet the credit policy. Status:Fully Paid" = "Does not meet the cr
                            "Does not meet the credit policy. Status:Charged Off" = "Does not meet the
                        )),
```

```r
          # Horizontal line ----
          tags$hr(),

          # Price Range filter
          sliderInput("dti",
                      "Debt to Income Ratio:",
                      pre = "%",
                      min = -1,
                      max = 999,
                      value = c(0, 100)),
          # Horizontal line ----
          tags$hr(),
          tags$div(style="display:inline-block;width:100%;color:#e48806",
                   "** Due to large dataset size, visualizations and filters may run slow **"),
        ),

      # Show a plot of the generated distribution
      mainPanel(
          tabsetPanel(
             tabPanel("Year wise loans trend", shinycssloaders::withSpinner(plotOutput("loanProcesse
                      shinycssloaders::withSpinner(plotOutput("totalFundedLoanAmountEachYear"))),
             tabPanel("Loan Amount, Term Relation",
                      shinycssloaders::withSpinner(plotOutput("loanAmtTermRelation", height="700px")
             tabPanel("DTI Trend",
                      shinycssloaders::withSpinner(plotOutput("dtiTrend", height="700px"))),
             tabPanel("Loan Amount, Income, Interest Relation",
                      shinycssloaders::withSpinner(plotOutput("fundedAmtIncomeAndInterestRelation"))
                      shinycssloaders::withSpinner(plotOutput("incomeTrend"))),
             tabPanel("Loan Amount by state",
                      shinycssloaders::withSpinner(plotOutput("loanFundedAmtByState", height="700px"
             tabPanel("Loans status and purpose",
                      shinycssloaders::withSpinner(plotOutput("loansByStatus")),
                      shinycssloaders::withSpinner(plotOutput("loansByPurpose")))
          )
      ),

   )
))
```