**Video Script: Section 6 Video 2 – understanding the structure of a shiny app.**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Description | Action on screen | Narration |
| 1 | Introduction  (Outcome and why it is desirable)  This should give the viewer an idea of the outcome of the task at the beginning of the videos and set the stage and expectations of the viewer. | Refer to PPT | In this video, we are going to study the structure of a Shiny app and see how each part of this structure interacts with the other. |
| 2 | Context(Problem/Solution)  Present the viewer with a real-world solution and how the situation would pose as a challenge. It always helps to draw the viewer's attention using a use-case. Metadata template can be used here. |  | If we want to build our own complex interactive webpages, we need to understand how the information is passed on from the browser to the R code and back. For this we need to know how a Shiny app is structured. |
| 3 | Guidance (How to do it and how it works): | Switch to Rstudio.  In the console:  Getwd()  On the screen: Section 07 | Open Rstudio and make sure you are in the folder ‘Section 6’ with either getwd() or by looking up the path on top of the console.  You can change your working directory from the console with the command setwd().  You can also use the 'set as working directory' button in Rstudio from the files panel. |
| 4 |  | In the console:  library(“shiny”)  runApp(“activity\_06\_02”) | Run the following command in the console:  library(“shiny”)  runApp(“activity\_06\_02”) |
| 5 |  | A web browser opens, with a page containing an interactive graph.  A description... | This is the example that we saw in the first video.  <play a bit> |
| 6 |  | Back to RStudio | Let's stop the server and see the code that generated this page. |
| 7 |  |  | The code sits in the folder “activity\_06\_02”.  The argument for runApp is the name of the folder. |
| 8 |  |  | Shiny apps always consist of at least two files, which are named UI.R and server.R |
| 9 |  | Open UI.R and server.R in the editor. | As their names suggest, UI.R (for User Interface) contains the code for what appears on the webpage: the text, the plots' location and the controls, while server.R contains the code that generates and updates the content of the webpage: the graphs and the results of a calculation. |
| 10 |  | Go to UI.R. highlight where appropriate | Look at ui.R  The code consists of a call to shinyUI.  pageWithSidebar provides the general layout of the page: a header, a sidebar on the left handside and the main panel. It produces a modern and lean web interface.  headerPanel sets the page title and sidebarPanel adds some controls in the side bar.  The mainPanel is populated by a single plot (plotOutput). |
| 11 |  | Go to server.R  highlight where appropriate | Now let's move on to server.R  server.R consists of a single call to the function shinyServer.  shinyServer always has two arguments named inputs and outputs, which are lists.  The information from the page is passed on to the server via the properties of input, e.g. input$obs.  In return, the server returns its results (graphs or text) via the properties of output, e.g. output$distplot. |
| 12 |  | Switch back and forth UI.R and server.R  highlight where appropriate | The names of input’s properties are given in the call to the controls in UI.R: input$obs in server.R comes from sliderInput("obs"),… in UI.R. |
| 13 |  | Switch back and forth UI.R and server.R  highlight where appropriate | Likewise, output’s properties in server.R connect to the rendering functions in UI.R:  output$distPlot in server.R is fed to plotOutput("distPlot") in UI.R. |
| 14 |  |  |  |
| 15 |  |  |  |
| 16 | Conclusion:The video concludes by showing the viewer that the goal has been achieved, and reminding them why they should be happy about that. A PowerPoint summary slide with the key points emphasized would make it easier for the viewer to remember what was covered in the video | Back to PPT | In this video, we briefly saw how shiny works: a file UI.R sets the webpage's layout , while a second file, server.R, computes the content. They communicate using two lists: input and output.  We’ll learn more about this process later when we’ll study reactive programming, in video 04.  In the next section, we’ll see how to generate textual content on a webpage. |