**Chapter 4: Design and implementation**

**4.1 Design**

**4.1.1 System architecture diagram and component descriptions**

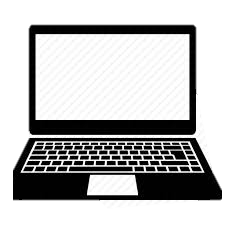
**User**

****

**Front end**

**Data store**





Laptop displaying app

User interaction

Web app (RShiny, R)



Queries app

Displays data

**Back end**



Web server (Shiny server)

Sends request

Sends data (graph data etc.)

Sends raw data

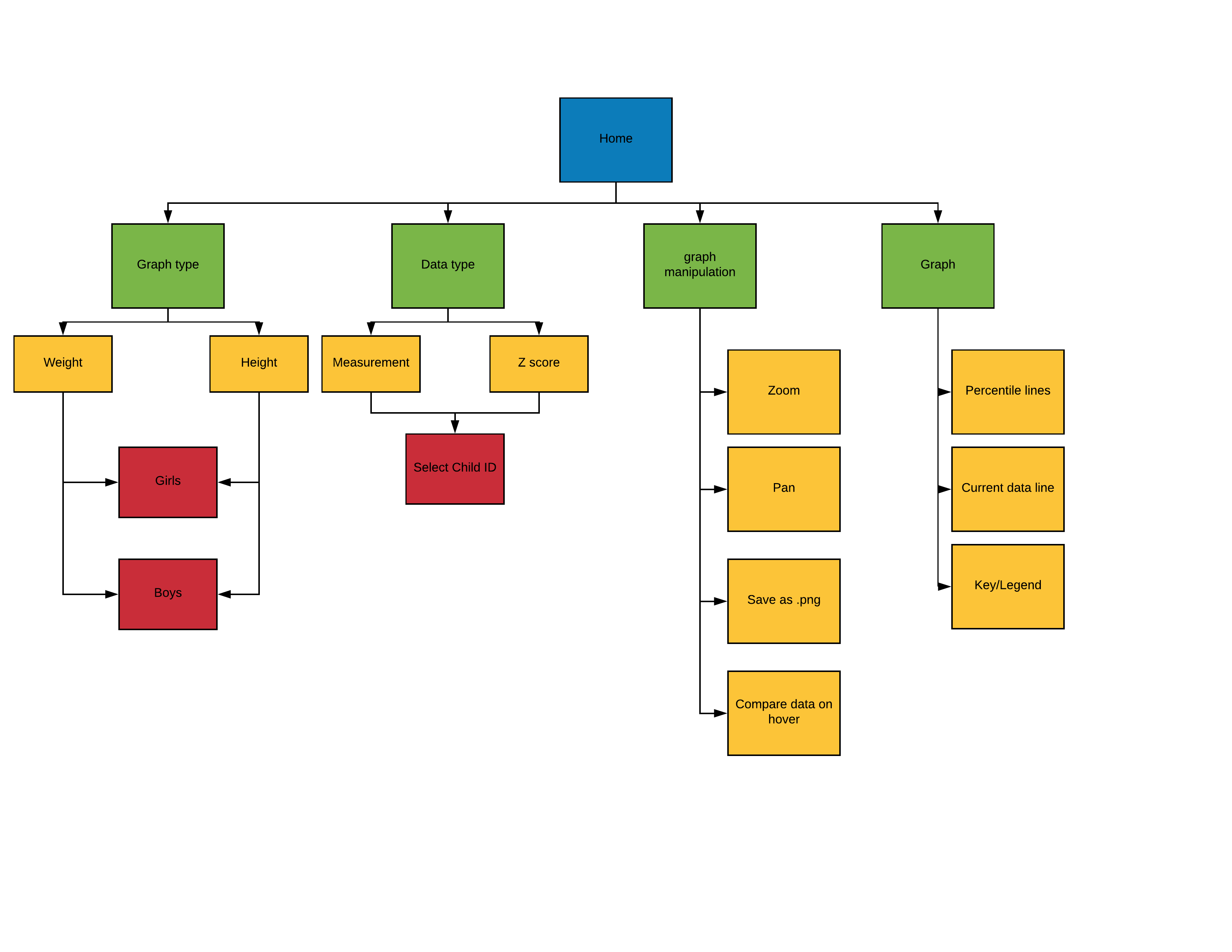
Requests data

Flat file of data of child growth (.csv format)

When interacting with the application, the user is presented with the front end of the system (the web app / laptop). From here, the user can request data such as the male height growth chart by clicking the relevant buttons and the data is also displayed here for the user to access. When data is requested by the user, a request is sent to the back end of the system (the shiny server hosting the application) which then retrieves the data from the .csv file uploaded to the server, the data is then sent up back to the web app so that it can be processed and outputted to the user accordingly.

The data store we used to store the data on child growth was a .csv file as there are built in functions in our front end, R that allowed us to manipulate .csv files to extract data with ease. We used “shiny server” as our back end as there is a module that can be added to R called ‘RShiny’ that easily allowed us to port our R code into a web app that can be accessed on all platforms without needing the sourcecode to be present

**4.1.2 Site map**

****

**4.2 Implementation**

**4.2.1 Version control**

While developing our project we used Git to share what we were working on with ease. Through use of this software we were able to simultaneously work on the code and commit it to a master branch when we want to use what we’ve created. This has also saved us from having to send constant emails regarding changes and having to trust one person with the code etc.

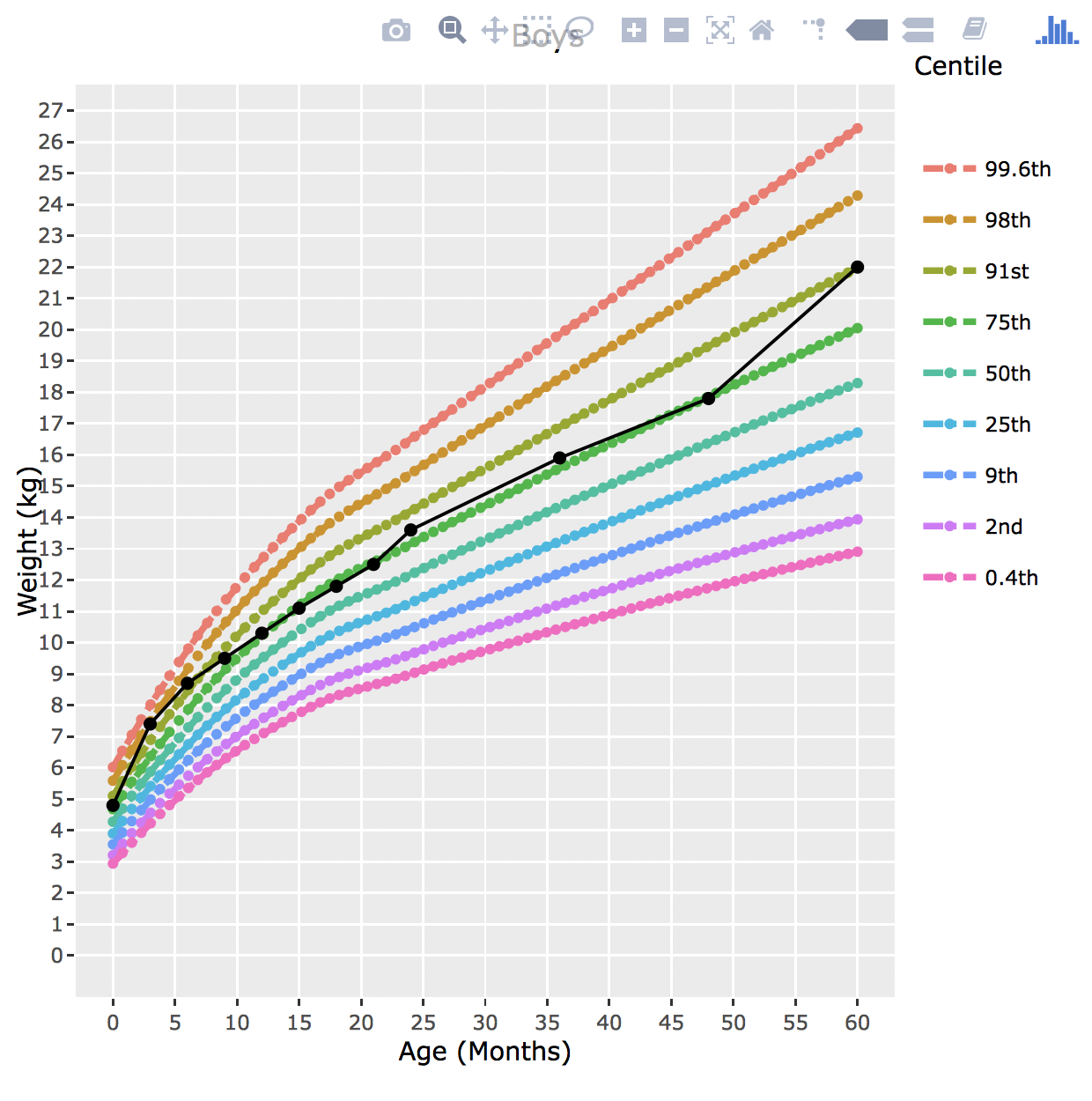
**4.2.2 RStudio**

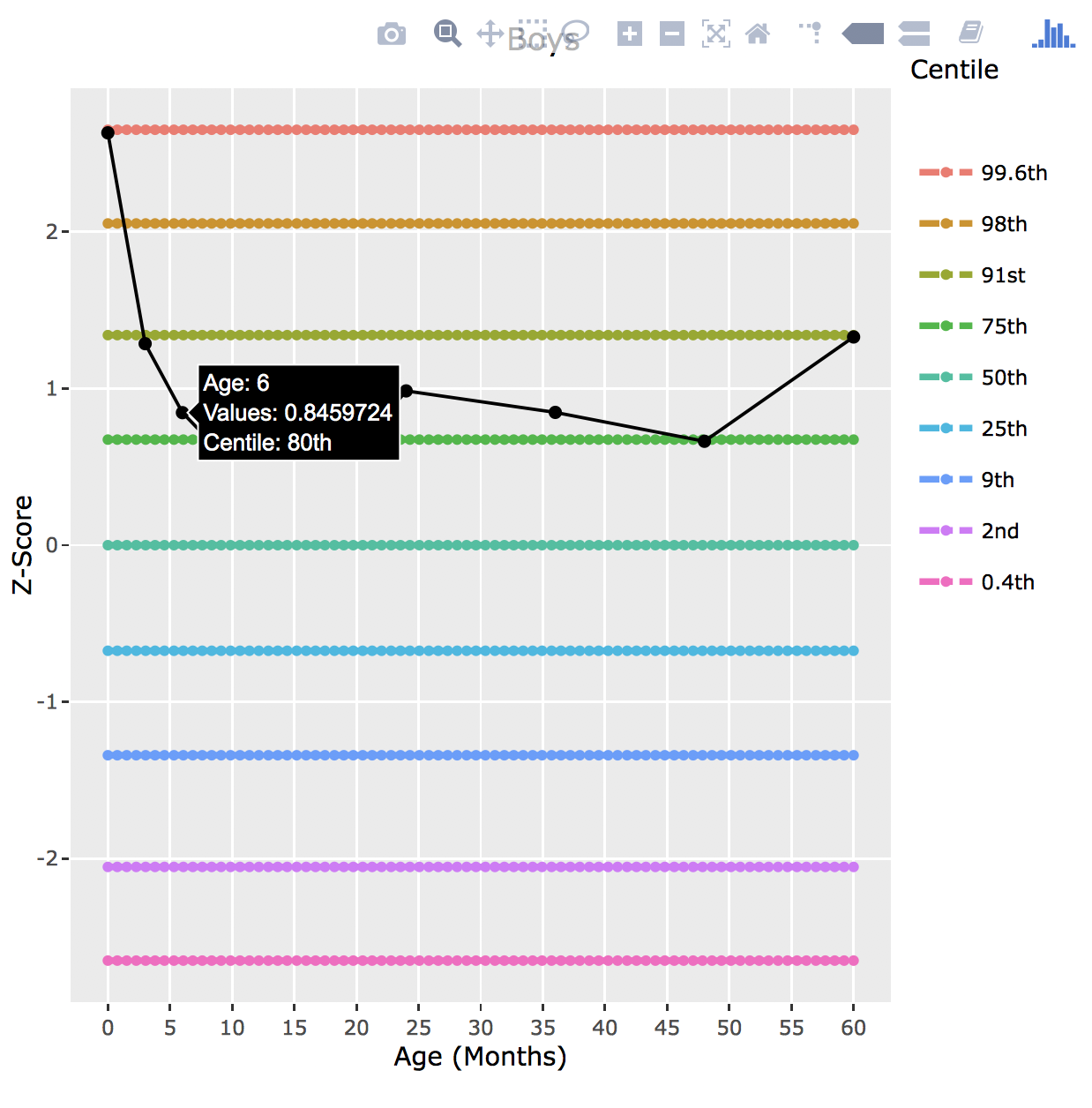
We used RStudio as our IDE for coding the front end of our application as it is the industry standard for coding programs in R and our client, Tim Cole advised us to use R for the creation of our application as it has very easy to use modules and functions made for displaying data on graphs which is what the main purpose of our app is.

**4.2.3 RShiny and shiny server**

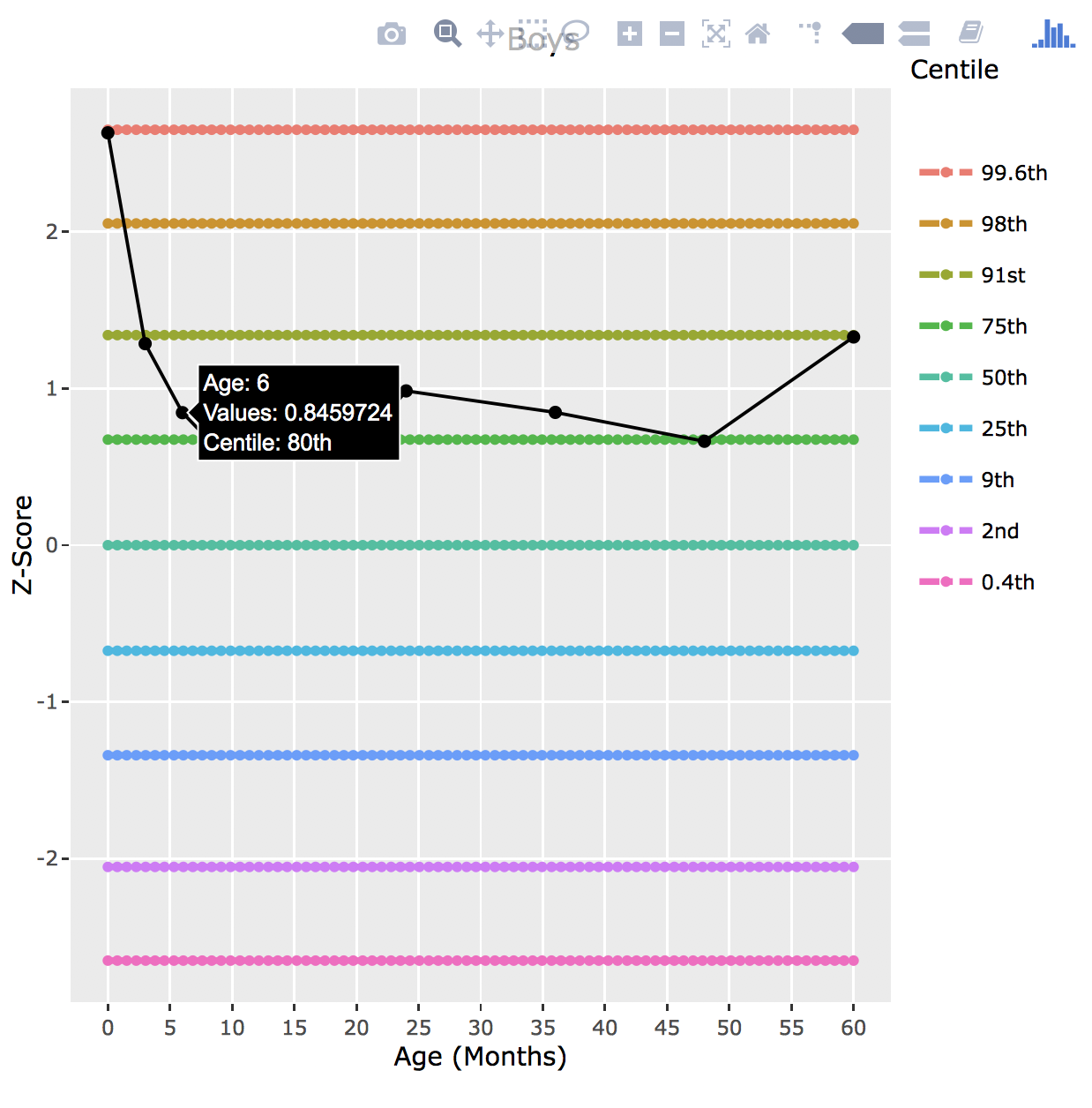
RShiny is a module we used for enabling our R code to uploaded to the net so that our app can be used as a web app that can be accessed across all platforms with web-based e universally accessed as well as acting as our back end, we used this server as it has direct compatibility with RShiny and is free to use.

**4.2.4 Implementation of core features**

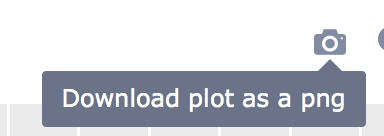
**Showing weight/height graphs with centile curves:** in order to implement this core feature we used a library in R called plotly to create a graph that is aesthetically pleasing for the user. We used this library over others as it had simple and easy to use built in functions that allowed us to add graph features such as a legend, axis titles as well as features that allow the user to manipulate the graph for example zooming in and panning onto different sections of the graph and being able to see the exact value at a particular point on the graph upon hovering. What we did to create the values for the graph was import all the LMS values from a .csv file into our R script and using an algorithm that our client, Tim Cole gave to us we were able to use the LMS values to calculate a measurement value that we use for the graph. For the different centile lines, similarly to what we did for each child’s data we had a set of values for each centile that we imported via a .csv and using the same algorithm we calculated the measurements for each desired centile and plotted each centile as a distinctly coloured line.

**Transforming measurement graphs into z score graphs:** In addition to having a measurement time graph for the user to see, we also added another type of graph that makes it easier for the user to compare a child’s growth to the national average more directly. The graph we chose to do that will achieve this is a z score time graph which shows how the child’s growth changes in the centiles over time. We did this by importing the same set of LMS values for each child that we used for the measurement time graph but instead used a different formula that calculates the z score instead of the measurement. This formula was also provided to us by our client Tim Cole.

**Viewing centile on hover:**

When we first created the z score time graph, the centile wasn’t showing when the user hovers over a point instead only showing the calculated z value at that point which doesn’t clearly indicate how that specific child’s growth compares to the rest of the sample. What we did to implement this feature was add a ‘centile’ column to the data frame which contained the centile for each calculated z value. With this, we then used a function in the ‘plotly’ library that includes the value of this column which comes up when the user hovers over a point in the graph. Although this may seem like quite a small feature, it is the feature we are most proud of as it shows the user a direct and clear comparison between a particular child’s growth and the rest of the general population making our application more advatageous and resourceful than the current method of viewing child growth charts (paper format).

**Saving graph to an image file:**



Through our use of the ‘plotly’ library in R, when we created a graph, there was already a built in feature within the library that allows the user to save the graph to a .png file. This is a useful feature as it would allow doctors to send a particular childs growth chart to the patient for their own personal use and without having to have our application.