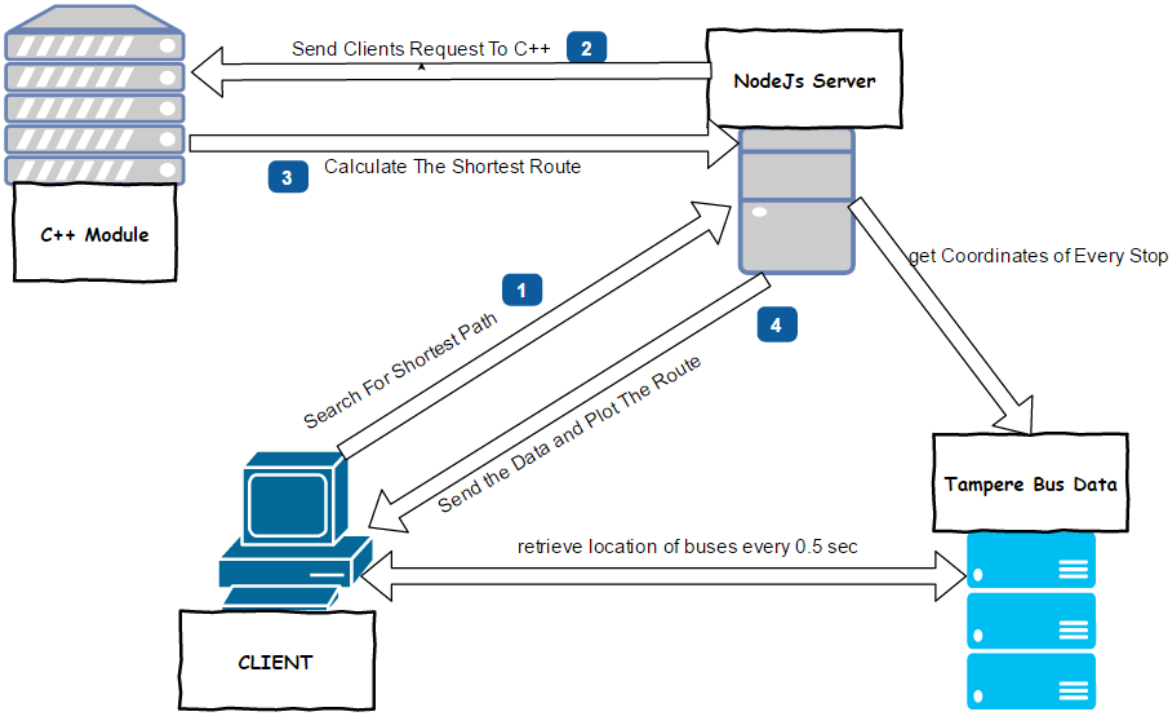
ARCHITECTURE



From my course work I was given zzz-final_bus_full.txt file, which contains all the information in order to calculate the shortest route from one bus stop to the other.

This file contains the following Information:

1. Stop names & ids.
2. Route names & ids.
3. Stops that lies on the specific route.
4. Bus names & ids and their times.

Preliminary work:

Since, I was provided with only the StopID's, therefore, to plot the route on MAPS, I needed the latitude and longitude of each stop.

Therefore, I used TAMPERE Journey OPEN API, and saved the corresponding stop data in the file location_stops_coordinates.txt.

FORMAT:

```
StopID, StopName, Coordinates
```

```
8222;Raskintie;61.49872,24.04527
```

Process 1: Client Request to Calculate Shortest Route

Process starts by a client request for a shortest route between specific locations. When the client clicks the SEARCH button, http request is sent to the NodeJs server. The server checks the parameters that are required for the algorithm to be executed.

The parameters are:

- Time (time user wants to leave from the Start Stop)
- Start Stop
- End Stop

Process 2: Servers Request to C++ Module

Node facilitates creation of child processes to leverage parallel processing on systems. I used "child_process" module and used the **exec** method which runs a command in a shell/console and buffers the output. As soon as Node receives the request it executes the C++ file.

Process 3: C++ Module outputs the result to Node

The format of output from the C++ algorithm is as follow:

Time; StopID; StopName

C++ executes the algorithm and returns the response in the above format. As soon as Node receives the output, it converts the output in the format required by the Client to plot the Route. Therefore, as mentioned earlier, Node takes the stop Ids and retrieve their corresponding latitude and longitude.

Process 4: Node Sends Response Back to Client

After processing the data, Node sends the response back to the client. The response contains the latitude and longitude of all the stops in the route, and the corresponding time the bus reaches the stop.

Plotting Route On GOOGLE MAP:

Description of all the major functions are described in the README file.

Problems Encountered:

- Google MAP API only allows to plot 8 way points while plotting a route. One solution was to pick only 8 points from all the stops, and plot the route. But this would not have captured all the details. Therefore, I used an Array to store multiple `directionsServices.` `All the stops are divided in to arrays of 10 elements. And result from them are combined to display the route, passing from every individual stop.`

- API does not provide a Click event on the route obtained. Therefore, its difficult to visualize the distance between two consecutive stops. In order, to add the specific functionality, I had to create a function which created poly lines on the given route. Thereafter, I attached a Click event with each of the Poly Lines. For the time being, the Information window only shows the distance between two stop, but one can also add directions on the click event as well.

- Firstly, I used PNG file to display buses moving on the route. But the problem with PNG file is that I cannot animate the PNG file. Therefore, I created an SVG path which displayed bus as pointed arrow. I added rotation property to the MARKER, and the head of the pointer indicates the direction of the bus.

Future Prospects:

- A separate SVG could be added for each bus, which will indicate the number of the bus on top of the arrow. Therefore, further increasing the monitoring process.
- A web server could be made using C++, which could be accessed via REST API. A user will hit the specific end-point, and algorithm will calculate the PATH and will send it in response.

Working Examples:

- Start Point : Keskustori M 0001
  End Point: Hervantakeskus 3642

- Start Point : Aleksis Kiven katu 0070
  End Point: Lapinniemen kylpylä 5018