Let me tell you something about my smart bike. I initially chose to make this project since I always go to school by bike and thought, why not do something with it? I ended up deciding to use sensors and actuators to add to a simple bike ride. I first tried to get all the sensors in order and tried to figure out what would be most useful to have and/or to know while you're going places.

So, I ended up using a DHT11, which measures both the temperature and humidity, alongside a light-dependent resistor and a hall sensor. In that way, I could also get the light intensity and try to measure the speed and distance traveled by the bike. My first struggle was figuring out how to actually properly measure or calculate the speed based on the rotations made by the wheel that the hall sensor could detect.

After I got this all in order, I knew I'd have to figure out how to properly store this data. So, I went on and set up my database using MySQL. The ERD was okay, but eventually, I had to change some things here and there to make it work with other adjustments I had made along the way.

There were a lot of moments where altering one part or trying to set up or simply troubleshoot something sent you all over the place, from frontend to backend to the database and so forth.

After I got most of these things in order, my next struggle was switching around my methods. I am using a Raspberry Pi and an ESP32. Initially, I had thought to send over files kept by the ESP to the RasPi, but what ended up happening was I made the ESP32 act as a little server, and the RasPi was then looking for those files. So, I had to switch my methods around. Of course, the file concept had to stay so you'd be able to gather up data as you'd be riding your bike. I then simply switched it around and figured out how to send a POST request from my ESP32 over Wi-Fi to my Raspberry Pi.

It was a struggle, and I almost lost hope there, but I finally found a way to properly receive it after messing with both sides of the code for a while. Once my data was there, I, of course, had to make sure it was able to fluently work with the rest of my code, and I didn't have to reformat anything.

The next thing on the line for me was my frontend. I had finally almost fully figured out my data, so now my frontend had to be worked on at least a little. I went on and found an image I thought suited the general concept and then tried figuring out how I'd want to display every little bit of my data.

I ended up splitting it up into 3 pages. The homepage displaying a general image, a small word about the project, a RasPi kill switch because why not, and also a map with the date and time of the last route you did. In all seriousness, the switch for the RasPi is there for in the case of you not wanting to have the data sent to the RasPi when you get home or for if you'd prefer to get some data nearby or something along those lines. This way, when you next get the RasPi going, the data will just transfer over all at once. Now, I did say 3 pages, so the other 2 were for Data and Routes.

The data page speaks for itself somewhat, and that displays 2 charts: one with live data and the other displaying the data from the last route you did. Below that, there's some info about the data, namely the minimums, maximums, and average values of your sensors.

The route page was a whole struggle of its own. I left this one empty for quite a while, working on the other things first. As you can imagine, figuring out a route or displaying one would require coordinates. Where do you get those? Well, those come from a GPS module in my case. So, figuring out how it worked, I went on and gathered some GPS data. Outside that is, these modules are actually not strong enough to get a fix on a satellite when they're not almost outside or outside.

Well, that said, I did get my data, and while I initially tried to just go straight for the data and figure out its accuracy and such, I had to eventually do the same without a library. A GPS module sends quite a lot of data, maybe more than you'd think. I got the latitude, longitude, and eventually opted for getting the time from there too since I had to switch my entire code for the ESP32, so Arduino code, over to also work without Wi-Fi. This included figuring out how to get time without Wi-Fi, which then sent me back to my GPS module. This required some messing with code and reformatting since I didn't get the date from the GPS, but overall, this was okay. After the GPS module and that side was basically done, I once again focused more on the frontend and somewhat the backend, figuring out how I'd store the GPS data and then, of course, how I'd display it.

I messed with some code in Python and got my coordinates from the GPS module onto a chart, which was very nice to see. The imports and things that I used for this were actually quite simple, and you could go about displaying it by simply feeding it the latitude and longitude coordinates. Since the coordinates are in degrees and minutes when gathered from the GPS, I did have to convert them to a straight value, but it went well. After this, I first displayed the last route on the home page of the project and then moved on towards the route page.

As I mentioned, I had avoided the route page for a while since I struggled to figure out the exact way I'd like to display it. I debated if I should give it another page to display the maps, and then I just went with something. I settled on a table, a table with a header and different rows. Something I might not have mentioned at this point is that when I say "frontend" and the different pages, it means I'm trying to figure out how to do things in HTML and CSS, and sometimes some JavaScript, which I used mainly to receive data I'd sent over from my backend and then display it. So, from this backend, I made some SQL queries that gave me the last timestamp of a given route, the route itself, and the last coordinates, and I sent all of this over to the JavaScript.

The table that I had made was now done... well, almost. I needed my maps. So, I thought, what if I could simply make a map pop out when you click on the route? And then, of course, as always, I struggled. Quite a lot, actually. First, there was the clickable row itself, which had to, of course, open. I made this happen with a click event and adding a place for the map to be, basically telling the JavaScript code, "When you click this row, please add a box here for the map and make it the same size as my table." Oh, and yes, that "please" was something I said a lot after trying out new code. It worked, and I thought, "Nice!" But then, I remembered I still had to put a map in it. After a lot of going back and forth and trying to send my data in a format that was good, I figured something out.

I found a way to do the table the way I wanted. Basically, this table was split into two pieces of data. One piece was for the table rows to say, "Well, this is the route, this is when, and these are the coordinates." However, if I wanted to display the entire route of that route ID, then, well, I'd need a lot more coordinates. So, I went back to my backend and started altering stuff until I had it working. I sent all of my data, well, not everything, but at least the coordinates of every route. Then, I managed to extract the specific route by comparing the route ID that was clicked to the others in the big bunch of data. This actually took me a while to get it done, but it worked, and I was happy.

At this point, I thought I had reached the end or close to the end of my project. However, problems have a tendency to appear when you least expect them, so of course, the data portion had to get a bit messy, especially on the side of my charts. After a day or two, I figured out how to solve it by altering the format in which my data was sent and making some small code alterations. Then I had to get ready for the real struggle, though.

As my project was finally seeing its final form, I knew I had to put it all together in my case, which of course, I had accidentally made a little too small. After a lot of struggling and very precise moving of every little piece, I got the project into its case. I decided to use the plastic of a U-shaped paper folder to cover the top of the 3D printed case, which ended up working fine. It was all very cramped, but I was able to check and it still worked.

So that was it. I hope you got some insight into what I had to do to create this, and furthermore, I hope you enjoy working with it or using it yourself.

Just let me know if you need any further assistance!