What makes distributed programming unique is the ability to have multiple computers work together, making it only seem like one computer. For our health-fitness application, to incorporate distributed programming, we thought about implementing a messaging feature between the coach and gym member. Due to time and resources, we were unable to work it into our project. Our plan of action was to allow users from different devices to communicate over a server. The communication between the coach and client allows discussion of goals, assign tasks to do, and give live updates on how the client is feeling and progressing. The ability to have a messaging feature allows the coach to check in on clients in real-time to see if they are staying on track.

Logistically, to implement this feature, the socket module would need to be imported to create a socket server. To represent communication between a coach and a client, it would be done with a server and client. A client and server program would be needed to create and show the communication over the network. The server program would have to grab a host name and create an instance with the socket module. With the ‘listens’ function it can specify how many connections. The client program is similar except no binding takes place. The user would input a message which would be sent over with the ‘send’ function in which the server would receive with ‘recv’. This would allow a conversation to occur until a key word is implemented by the user that would terminate the connection.

When looking at the possible containerizing applications, there were many to consider. In selecting a potential containerizing application, we decided we would do Google Cloud Run. One of the reasons we chose it is because of the simplicity it offers for deploying a program. The way it works is that we create our program and package it up in a container and run "gcloud run deploy" which would take it live. The application works with many languages including Python,

which is an additional benefit. It automatically scales and gives a free trial to test out the application.

For our program, once it were to go live we would run our application from a different device to initiate the messaging. This is how we would implement the distributed portion of our program.