Question 1

- **GitHub Repo Link**: https://github.com/hjk216/aura-provider-portal
- **Repo README Contains**: Summary, instructions, assumptions, next steps
- **NOTE**: Change to data collected (if this was real). One potential issue in the patient data collected is the age. The source data should really be date of birth since age will change every year. Can always display the patient's current age on the client.

Question 2

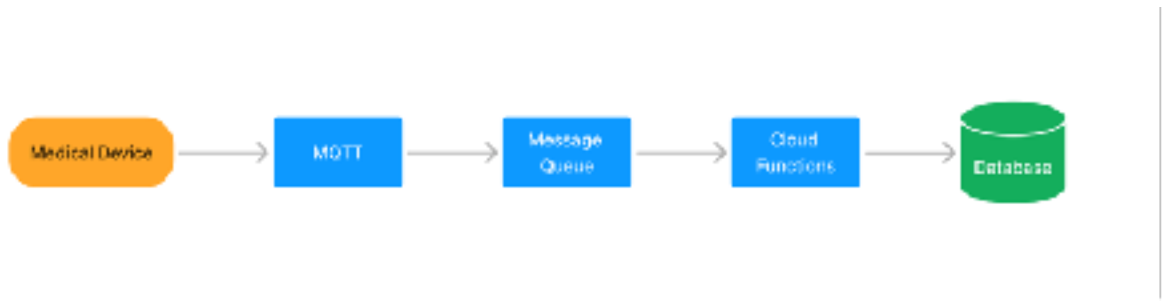
1. **Regulations & Choosing Database**: Since our target market is the US, verifying that we are storing our patient data in compliance with HIPAA and any other relevant healthcare regulations. This may effect which database or which services to use.
2. **User Management**: Implementing user authentication and authorization is essential. There are many tools we can use for this including Auth0 or any major cloud provider's solution.
   1. On authentication, basic user account management and perhaps requiring MFA (multi-factor authentication).
   2. On authorization, especially when dealing with sensitive patient data, the "principle of least privilege" should be used here to allow providers access to only the necessary patient information.
   3. This also requires creating the necessary user models on the backend.
3. **Testing**: Writing tests for the back and front ends. If using Django there are Django unit tests, for React could use something such as Jest. Even though reading and writing simple data is straightforward as the application evolves testing will become more important.
4. **Logging**: This may not require any action but thought I'd include it. Many hosting solutions provide logging by default. We may want to use a specific logging solution depending on what sort of events we'd like to track. It's also important to note here that we don't want to log any sensitive information as that could break regulatory compliance.
5. **Other**: Choosing cloud hosting providers and services, setting up CI/CD, and configuring the settings for the back- and front-end frameworks. Also, could use better validation and error handling on the client side.

Question 3

1. The key words here is "real-time information," which in the context implies to me that we will be processing a large amount of information relatively quickly.
   1. We could use a pub/sub solution like MQTT, especially if connectivity could be unreliable, send this information to a message queue such as Kafka or GCP equivalent, then to our database.
   2. Depending on how we are using the data we may consider aggregating or batching the data to avoid excessive writes to the database. We could include cloud functions in order to complete some of these tasks.
   3. Evolutions: could add more consumers to the message queue for other services such as notification services or analytics services.

    4. Diagram below.
2. I don't have much experience or knowledge of GCP services, but I can tell you my general thought process.
    1. I would suggest using managed, automatic scaling based hosting services that still give us flexibility if need be. As a small team, focusing on delivering the initial product is the top priority rather than focusing too much on infrastructure.
    2. In general, the services that would be needed for this system: pub/sub, message queue, cloud functions, database. We would also be using services for logging and using Google IAM.

Question 3 - 1: Design Diagram


Medical Device
MQTT
Message Queue
Cloud Functions
Database