Imprimed System Design

Front-end

React/Angular or other front-end framework is used with that assumption that thousands of lines of javascript code is used per http response. Preferably React because since we are also building mobile applications - learning curve for React Native is lower if the developer has working knowledge with React as opposed to other front-end framework.

Back-end

Our app requires machine learning and needs a lot of math computation therefore, python is the most appropriate language as it has great libraries for machine learning and math/statistics. With that being said, Django is chosen because of its MVC framework - which allows a faster development process in which a small team is able to work on their own projects without overriding each other's code.

Database

Since we are working with sensitive patient information (assumption: massive data), I would consider using Amazon S3 to keep our local database storage as lean as possible for speed and efficiency in data retrieval where large data will be stored at Amazon S3 or in any other competing cloud products and store the link to each large data onto our local database. In terms of managing data, I would choose SQL over NoSQL as it is more appropriate to be ACID compliant to reduce anomalies and safeguard data integrity for our data scientists to efficiently use data to produce better outcomes. Having said that, PostgreSQL is an ideal database system choice because it is highly ACID compliant and scalable to work with a massive database table. In next 3 to 5 years, if the data will be less than 1TB using PostgreSQL will be more than sufficient to handle the load, however if its more than 1TB and less than 16TB , I would consider using Amazon RDS (more specifically Amazon's Aurora) or any other competing cloud products, anything more than 16TB, I would strongly consider using Apache Hive.

Testing/deployment process

In my current job, I personally follow a similar approach to GitFlow where each developer is given separate repositories forked from the main/origin repository. So for continuous integration, each developer works on their own task and submits a pull request and the engineering lead reviews the code and rejects/approves the changes and approves for production. I would include Jenkins to help facilitate CI and CD especially for testing.