Scenario 1:

When building a healthcare platform, privacy and security must be top priorities. To support online appointment scheduling and calendar synchronization, we can create a custom calendar and integrate it with a unified calendar API like Nylas. This will allow seamless sync across all platforms.

For secure and reliable video conferencing, the Zoom SDK is a strong choice. It's easy to implement and offers a familiar interface, making it accessible even for older users.

We can have a series of prompts to get some initial data for the problem that the patient is facing then it can be fed to a Python program which can use different algorithms. We will need to have a dataset where it contains clinical texts like notes since we need to have natural language processing, symptoms and its diagnoses which can help in diagnosing condition from prompts.

To store and transmit the sensitive information, we can use AES-256 encryption and store the data on AWS with Server-side encryption as well. Every request should be authorized and authenticated and logged. To comply with HIPAA and GDPR laws there should be consent of patients to store the data and the right to access/erase be provided to patients. To ensure system uptime we can host the platform on AWS with Load balancing and Auto scaling to ensure smooth experience even in high traffic situations. To handle user identity verification, we can have the user enable two factor authentication for enhanced security. Access control can be implemented by assessing the sensitive data and ensuring that each role has access to that data according to the regulations. This can be done via Role-based access control which restricts data based on user role.

Scenario 2:

Since we are building a news aggregator that uses machine learning, the most optimal language to use is Python, since it has great support for machine learning algorithms. The first thing that needs to be stored is the user logs which track the behavior the user and categorizes it based on domain. Then this data can be fed to recommendation algorithm like content-based filtering algorithm which can recommend similar news that the user likes. To find the content we need to design a web crawler. Functional requirements of web crawler include url discovery, content extraction, prioritization and handling all types of content. Once the urls are discovered from various sources, we can extract relevant information form fetched pages, its content and metadata. Further this information can be prioritized based on various factors like relevance, time posted etc. Using this information and the behavioral logs sentiment analysis can be done to get the hot topics detected.

Google translate API's and cld3 can be used to manage multi language detection and to scale it even further we can host it on any of the cloud provider which can guarantee uptime and scalability. The first thing is the user consent for collecting data from their browsing history. Once that is obtained, we store only the categorized data of their history and not their personal data. To ensure fast article delivery we can store the most frequently accessed news articles and the latest news on redis cache. We can also store the cache based on user-level where we store the top recommendation for the user.

Scenario 3:

Since realtime tracking of data is needed and scalability is required. We can containerize the application and deploy it on AWS with load balancer and Auto scaling to ensure performance during high traffic. The data of the tracking can be stored in a time series database, since it accurately stores all the data and associates it with timestamps we can have very accurate live tracking and historical data. The google Maps api can be used to ensure accurate directions and dynamic route optimization. Since we need to support for multiple OS on mobile devices, the best technology is Flutter right now, since it supports cross platform development. Web dashboards can be made through react to make a single page responsive application ideal for tracking and the API's can be hosted on AWS as previously said, with support of web sockets to ensure real time updates. Using GPS tracking data we can find out the usage metrics of the vehicle, like the miles covered in what amount of time, last serviced etc, and this can help in predicting regular maintenance requirements. To ensure security we must have JWT based access to ensure secure sessions for both mobile and web apps. In order to ensure working of application in lower connectivity region, the nearby maps can be cached constantly and as soon as the network goes down the application should switch to cached maps, and when the connectivity is restored the data then can be transferred back to the server.

Scenario 4:

The data model for product can have the most common attributes stored in a map like color, size etc. Then the user can be given option to add custom attributes which can be directly added to that map which ensures easier and faster retrieval. We can have a microservice based architecture, where we can have a microservice for each component like user authentication, products, listings, scam detection etc. This makes it easier to scale each component individually and eases deployments and reduces downtime. To ensure fairness and scalable review system, we set up a few rules where only verified users and users who have purchased the product can review a product. We can also make it such that we can allow other users to upvote and downvote a review. These reviews can be stored in a nosql database since it allows fast access to reviews and it does not have a specific format of storing data, it will make it easier to restructure the reviews. These reviews then can also be used to find potential frauds and scams. The scam detection module can constantly monitor all purchases and reviews to find anomalies and report potential scam products to the moderators for further action.

To ensure secure communication between buyers and sellers we can use End to end encryption of messages between them. To give estimated shipping costs we can give the sellers two options, where they can handle shipping by themselves and give estimates based on weight, distance, zones where they deliver to and using this we can calculate an estimate of costs. Another option is to use a third-party shipper API like fedex, where we can just feed the information of pickup and drop off and they can give an estimate of their services. For image optimization we can use ImageMagick which can easily perform conversion on image and make it compatible with out website with proper dimensions. When the image is processed it can be stored in amazon s3 bucket which allows for easy and fast image retrieval with good uptime and data security.