Task 2: Written Answer

To make sure our API can handle increased traffic without fail, and can scale with the increase in number of users, multiple steps can be taken viz.:

1. Database optimization techniques
   The most important form of optimization can be database optimization, as this is the first issue faced in a growing application. We can use indexing to make sure the access to a large database is fast and reliable, without needing to go through the entire database over and over again for every query. With this, we can look at a small subset to find our required results quicker. We can use multiple levels of indexing if needed.

2. Caching strategies
   The second optimization we can use is caching. Caching will allow us to locally store a small part of the database the under heavy read load in a local buffer (or faster access memory), which can allow us to serve these results quickly, without ever consulting with the database. Although if we apply this, we need to invalidate the cache if there is an update within the data, to make sure latest data is provided without fail.

3. Asynchronous processing

We can use asynchronous processing techniques, to handle tasks can be offloaded, and don't need to happen during the processing of the application. An example could be putting off sending a welcome email to a new user in a task queue, which is only served when the server is not under heavy load, thus allowing for quicker responses to queries that are more important to serve in real time.

4. Load Balancing and horizontal scaling
   Instead of making one server powerful, to handle multiple requests, we can instead use multiple servers, that are distributed either according to geography, or load (or both), to handle multiple requests smoothly without tanking the processing speed of a single server unit. For this, we can use a load balancer service that redirects the users to different servers, based on load and where they are located. Needless to say, we need to make sure the database is located centrally, so that everyone can access the latest data, not just some part of it.

5. Efficient use of serializers and Querysets
   To efficiently make the use of the features provided to us by Django, we can write queries to make sure only the required data is pulled from the database. For example, if we only need to return email and username of a user, we only return that part of the data, making the process faster, and less cluttered.