Thank you for the opportunity to work on the URL Shortener project. Below is my analysis of the requirements, possible implementation options and the rationale for my chosen approach.

Understanding the Requirements:
Let's assume our system receives 1000 requests per second, and we want the shortened URLs (slugs) to be valid for no longer than, let's say, one year. This means we need to keep:

1000 • 60 • 60 • 24 • 365 = **31536000000** (approximately **31.5 billion**)

mappings between actual URLs and slugs.

Determining Slug Length for Shortened URLs:
Since the shortened URLs should be as short as possible to be more "user-friendly," and the slugs are alphanumeric (containing digits 0-9 and lowercase and uppercase letters a-z, in total 62 characters), we need to determine the appropriate slug length (n). Using the formula $62^{(n-1)} < 31536000000 < 62^n$, we find that $n = 6$.

Implementation Analysis:
There are several options to implement this functionality within the given constraints, but we will face a dilemma no matter which one we choose, as each option has its own advantages and drawbacks.

Option 1 (Sequential Slugs): We store the slugs as sequential values (e.g., abc123, abc124, abc125, ...).
- **Advantages:** Easy to implement, with no additional hashing or mapping mechanisms required, as well as without any collisions.
- **Drawbacks:** It is easy to guess the next slug, which may be unacceptable for certain use cases.

Option 2 (Pre-generated Slugs): We store all the possible slug values in the database and randomly assign one to each URL.
- **Advantages:** This approach eliminates the deterministic method mentioned above, and also makes collisions impossible.
- **Drawbacks:** We would need to store 31.5 billion records in advance, which would require a significant amount of storage. Additionally, we would need extra layers and mechanisms, such as caching for fast retrieval and transactions (to ensure that two simultaneous requests are not mapped to the same slug).

Option 3 (Dynamic Slug Generation): We generate a slug each time and check whether it's already in the database.
- **Advantages:** Easy to implement, though some random data generation mechanism is required.
- **Drawbacks:** There is a risk of collisions, which would become a significant issue as the database grows. Additionally, this option does not address the need for implementing extra mechanisms, such as transactions.

Chosen Approach:
For this test assignment, designed to be completed within several hours, let's stick to the **Option 1 (Sequential Slugs)** due to its simplicity and ease of implementation. This approach allows for clear focus on core functionality while staying within the scope of the assignment.

Final notes:
Please note that I am not addressing scalability issues or database optimization aspects. With some great optimism I'm looking forward to discussing all the further optimizations and improvements during the next stage of the interview. Thank you for considering my work!

With regards,
Roman