Answers:

Q1: How can we enhance the backend to have faster performance for frequently accessed “short URLs”?

A1: We can enhance the backend performance by adding caching into the model. We can save entries in dict with LRU (Least Recently Used – discard the least accessed URL) logic, hence we save time by accessing the cache before accessing the database.

Q2: How can we enhance the implementation to have one “short URL” redirect to a few “long URLs”, based on redirect percentage?

A2: We can achieve the enhancement by adding a table to the database with the long URL as key, values of short URLs. We fetch the short URLs from the long URL and give them ids from 1 to the number of short URLs the long URL has. We randomly generate probability for each id (total of all id percentages is 100%) and randomly pick one short URL by percentage was given. This is for random probabilities. If the probability is high as much as got redirected, we can make the probabilities “bias” based on redirects field we add to the table.

Q3: How can we enhance the implementation to report the top 5 redirected URLs in the last hour?

A3: we can enhance the implementation by adding a field of counter (of redirects) and a time stamp to the URL’s table and adding a SQL query that fetches 5 (configurable size) entries from the last hour, with max counter.

Q4: How can we prevent an attack, in which a malicious client fills the database with millions of URL redirections?

A4: we can prevent attack by extending the solution from Q3. We can set a threshold. If the redirects counter incremented over the threshold in the last hour, we ban/block the malicious client.

Q5: What happens after the user presses the “Create short URL” button? Please detail as much as possible.

A5:

a. URL from input is sent to the ‘http://localhoust:5000/api/shorten’ (axios() function) for the server.

b. The server handle the call from the client by flask framework @route(‘api/shorten’) and call to the function shorten_url()

c. on shorten_url():

- c1. Validation of URL. If not valid send to database as “bad” and return to the client “not valid URL”.
- C2. If valid generates short code and then the short URL from the code.
- C3. Insert to database the code as primary key and the long URL as value (for redirect) and finally return the short URL to the client.