**Architecture/Software Design Test**

A client requires a URL shortening service, but has rejected using commercial SaaS such as bit.ly or tiny.url for cost or data protection reasons.

The shortening service should support the following functionality:

To shorten a URL, use a ?shorten query string; for example, <http://www.example.com/?shorten=http://some.url/etc/> will return a shortened URL <http://www.example.com/jhakdj>

To expand a URL simply request the shortened URL e.g. [http://www.example.com/jhakdj](http://b.io/jhakdj) and the system will redirect to the expanded URL.

To view the expanded URL without redirecting to it, simply add a + to the end of the URL, e.g. <http://www.example.com/xhgjgh+>

The development team who will build this system are experienced in Java; however other languages/frameworks would be considered with a suitable explanation.

The system is expected to receive around 1 million hits each day, with peak traffic of 50 hits per second.

Ideally the ‘unique code’ part of the URL would be no longer than 6 characters long.

This is not a coding exercise. For this test the candidate should produce a design or plan, which includes information/reasoning about the following:

* Suggested high level classes
  + and/or software frameworks/libraries
  + and/or any software design pattern
* how the data would be persisted
* how the system could scale and balance load
* Suggested server(s)
* Approximate server/hardware/monthly costs if hosted by Amazon AWS or an alternative hosting company from which it is possible to get costs

As the candidate is unable to ask questions of a business analyst or the client, any significant questions important assumptions should be documented.

**Solution: (Java 1.7/1.8)**

**Developing the API:**

URLService is a **RESTful** webservices application for URL shortening service developed using **SPRING** framework**.** The REST Controller will have RequestMapping annotations to call the handlers method that accepts ‘shorten’ parameter.

**How unique code is created?**

As per the requirement the unique code should be 6 characters long, at the maximum.

The unique code can be created using the following characters:

* a-z
* A-Z
* 0-9
* Symbols except special characters. Exclude ‘+’ as the requirement is use this character to view the expanded url.

While generating the unique code it is important to make sure they are not duplicated. Checking the database first for the url entry, and then generating the code is a costly process. Instead, we could make sure that a particular url always returns the same code. Encoding and salting the ‘shorten’ param value will always give the same result. First 6 characters of the result of MD5 hashing can be used. Should there be an conflict, i.e., same short code is generated for 2 different urls, the next 6 characters from the MD5 result can be used.

**Java classes created to implement this:**

|  |  |
| --- | --- |
| CodeGenerator.java | Reads the url from ‘shorten’ parameter and generates the code by performing MD5 hashing. This class could be a singleton, to keep the process of generating ids thread safe. |
| URLCodeStore.java | This is the data store class that interacts with database and performs retrieve, save, update operations. |
| URLShortenService | It’s the primary class that carries over the shortening works using the above 2 classes |
| URLServiceController | This is the entry class for the application that reads the url and maps the request to corresponding handlers. |

***Database considerations:***

In-memory databases like Reddis, Aerospike are good choices, as it reduces the hard disk usage.

Columns:

|  |  |  |
| --- | --- | --- |
| id | NUMBER | Auto-incremented column |
| code | VARCHAR(6) | NOT NULL, UNIQUE constraint |
| url | VARCHAR(255) | NOT NULL |
| timesCalled | NUMBER | This column can be incremented everytime the url with is called. This is just for future reporting purpose, in case if we want to see how popular a website is. |
| updatedTime | DATETIME | use getTime() everytime an update happens (or equivalent getTime() function) |
| createdTime | DATETIME | use getTime() when the row is created. This column is never updated again. (or equivalent getTime() function) |

Important things to note:

* The table that’s used to store the shorten url and full-length url must be **indexed**. So retrieval is faster. Using UNIQUE constraint would achievet hat.
* From within the application the developer should make sure the response can be built in single stored procedure call so the database trips are minimised. Might also consider using **JPA ( Hibernate**) to make use of its Caching capability so trips to databases are minimised to some level. This is if the in-memory data structures were not used.
* Increase the connection pool size for better performance.

**Tighter Security consideration:**

If security needs to be strict, for e.g., if the url contains highly sensitive information like Patient’s hospital number, NHS Number, treatment name etc

[http://www.example.com/?shorten=**http://some.url/etc/**](http://www.example.com/?shorten=http://some.url/etc/)

the ‘shorten’ url parameter (highlighted in red) may be an encrypted value, say an MD5 (or SHA256) Hashed token. Now, URLService must pass the token (URL value) DES encrypted value of a previously agreed key, and query for the name that’s stored in DB table as MD5 hashed and salted. The value against this name is the url <http://some.url/etc/>. Additional security can be added by deactivating the token right after first use.

Now, the shorten url can be generated as explained in the early part of this solution.

**To View or Open the url:**

Another request method should be written that accepts this shorten url parameter. Upon scanning the shortened param, if the request ends with ‘+’ , hit the database (in of the 2 routes explained previously) to get the full length url and respond back to the caller. Else, redirect the user to the new url using response.sendRedirect(newURL).

**User Interface:**

For **Responsive** User Interface, Angular/TypeScript languages can be used.

**Server Constraint:**

Consider having dedicated server to host this application as heavy traffic is assumed. **Tomcat** application serveris will solve the purpose of this requirement. (If not using AWS)

Tomcat can be configured such that each server instance can use a CPU core, in some cases, it can also use more than one core. Application servers should be **load-balanced. (F5 Loadbalancer).**

SSL enable the server to make sure the requests are coming from authenticated clients (Only if it’s absolutely necessary). But if validating the certificates are too much overhead for this requirement, we can make use of Server white listing features using Remote Address/Host filtering.

**Load/Performance testing:**

Performace testing softwares like **LoadRunner**, **JMeter** can used to mimic the bulk requests, and response time can be gauged. JMeter can run directly on AWS instances and the results can be verified by increasing/decreasing the number of instances.

Time the application/screen loading. If the loading seems to be slow, look for

1. Deadlock, thread loops using JVM thread dumps
2. Check Database connection pool size.
3. Increase app server instance and configure them to use dedicated CPU core.

**AWS Hosting - Prices**

The on-demand pricing for hosting the app in AWS is detailed here:

<https://aws.amazon.com/ec2/pricing/on-demand/>

By choosing on-demand, the client only have to pay for the EC2 instances that are used at a particular hour.