**URL app**

**Description**

Most of us are familiar with seeing URLs like bit.ly or t.co on our Twitter or Facebook

feeds. These are examples of shortened URLs, which are a short alias or pointer to a

longer page link. For example, I can send you the shortened

URL <http://bit.ly/SaaYw5> that will forward you to a very long Google URL with search

results on how to iron a shirt.

**Requirements**

* Design and implement an **API for short URL creation**
* There should be some form of persistent storage, but don’t waste too much time on database configuration **– simple file holding records or transitory runtime object is enough,**
* Implement **forwarding of short URLs to the original ones**,
* Assume application will **be distributed as Docker image**. **Provide Dockerfile**, but don’t waste too much time for building and testing docker image, **focus on functionality**.
* Assume importance levels:  
  1 – Code working as described in requirements,  
  2 – **Application** is **building** with simple javac, **mvn install** or gradle build command (or any basic build command working on behalf of programming language you choose),  
  3 - **Unit tests are included.** Coverage level depends on time you have left to complete the assignment, but we would like to see business logic **(service layer) coverage at 60%**,  
  4 – Other things you would like to implement for this project (ex. Database, application test coverage over 90%, API for gathering different statistics, UI or whatever else you think would make your application extraordinary),

**Assessment**

Treat this as a real project. It should be readable, maintainable, and extensible where

appropriate. The implementation should preferably be in Java, however any language can be used.

If you will transfer it to another team - it should be clear how to work with it and

what is going on. You should send us a link to a Git repository that we will be able to clone.

**SOLUTION IMPLEMENTED:**

1. Designed and implemented an **API for short URL creation using spring boot as rest end points.**
2. Used MYSQL local instance to connect to db , ***while running in local please change my sql properties in application.properties*** file to run in local machine.
3. As of now this done in back end (Through rest end point not implemented in UI) , I am attaching the video , which will get the original url by using the post man.
4. There will be two folders one is ***urlapi* which is complete back end** and the other one is **front end folder which is *urlapiui***  , provided docker for the both UI and Docker seperatlelt and docker-compose file for running all images together .

Provided the docker-compose file too, which configured all the images properly.

1. **Unit test cases are covered in service layer up to 85%**

**Path for this file generated is in E:\Home\_Assignment\_Work\urlapi\target\site\jacoco**

**The red one marked will vary according to the local system where you have the project.**

**Index.html file contains code coverage report.**

1. **UI implemented using React JS, only two files modified as of now just for entering the URL and convert it and show it back.**

**Note : included a short video of around 1 min.**