# Site Reliability Engineering – Practical Exercise

Thank you for applying for Site Reliability Engineering at Pearson Lanka. While we think that you will make a good fit to our team, this exercise will help us evaluate your level of expertise on different technologies, which are used in our day-to-day operations.

This document enlists the set of activities that you need to complete with the deliverables that you need to provide. It is expected that you use the most optimal solution for the exercises provided.

In case you have any concerns/issues in the assignment, please feel free to reach out to tharaka.perera@pearson.com.

After you complete the exercise, please forward us your solution – you can choose to host them on GitHub or any other version control system of your preference, however, please ensure that the repository is not public.

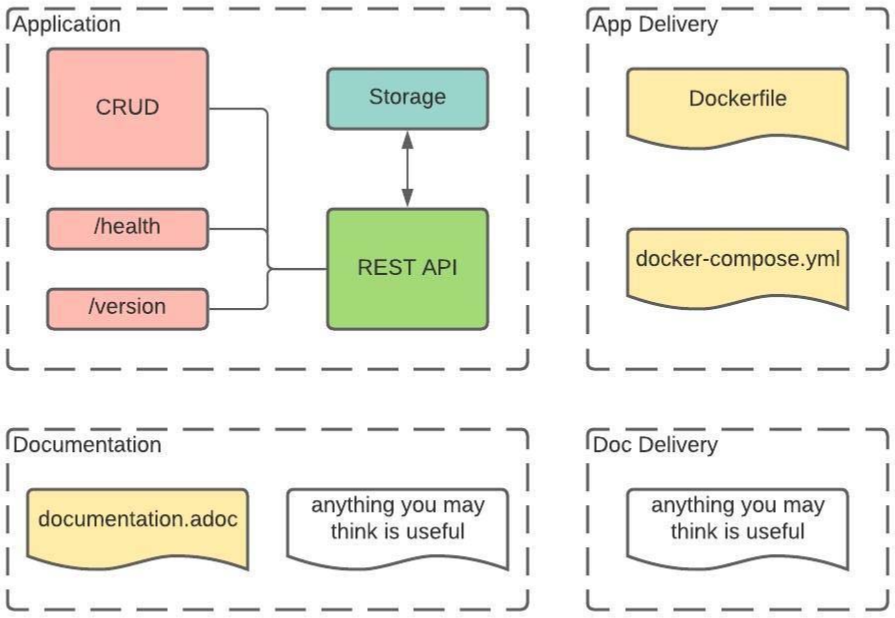
## Learning outcomes

Upon successful completion of this assignment, you should be familiar with following.

- Explain REST API services communicate with the backend server.
- List and discuss the basic CRUD functions.
- Recognize how to create GET requests to fetch row data and entire records.
- Identify how to create POST requests to insert data in databases.
- Identify how to create PUT requests to modify saved data.
- Recognize how to create DELETE requests to delete rows and entire records.
- Security best practices.
- Docker as a containerization technology.
- Source controlling and coding.
- Documentation standards.
- Infrastructure as code (IaC).

## Case Study

You are hired as an SRE (Site Reliability Engineering) for a major education service provider. Your manger has brough forward a stakeholder requirement to do a POC (Proof of Concept) of a simple application which will register users for upcoming online examinations.



## TASK 01

For the setup shown above, it is required that you script a CRUD API application using any of the following languages:

- Python
- NodeJS/Angular
- Java OR
- PHP

Feel free to make use of any light-weight database for your implementation (E.g., SQLAlchemy).

Requirements:

- Application should run at port 3000.
- API (Application Program Interfaces) output should be JSON (JavaScript Object Notation) format.
- All 4 operations should be present (CREATE, RETRIVE, UPDATE, DELETE).
- API should have responses from **`/version`**, **`/health`** URLs as well.

## TASK 02

Your application code should be stored in a development friendly platform and should utilize a standard DevOps Build tools to build the artifact. It is expected to containerize the above application with best practices which can be deployed across multiple environments.

- Use a suitable hosting platform such as Amazon Web Services free tier to host the above application.
- The APIs (Application Program Interfaces) should be accessible with a public Ip.
- Use IaC (Infrastructure as Code) tool for the infrastructure provisioning such as VPC (Virtual Private Cloud), Subnets, EC2 etc.

Requirements:

- `Dockerfile`
- `dockercompose.yml`
- Documentation in ASCII Doc format (README, etc.)
- Your application should contain the configuration file (e.g. -: database details, credentials).
- IaC should be scripted in Terraform or CloudFormation.

Optional Requirements:

- Use security best practices to store any sensitive data.
- Implement a login functionality to the application.