# Multiple choice question (15%, 10min, 10 questions)

# Scenario based short question 1 question

**CI/CD**

What are the key difference between **Continous Development** And **continuous deployment**

How would CI/CD improve the quality of software application?

# Scenario based short question**S** multiple related questions

- Version control

# Scenario based short question**S** multiple related questions

Automatic build processes

# Scenario based Long questions multiple short questions

## Recommended to use bullte points

Gradle?

# Scenario based Long questions multiple related questions

## Recommended to use bullte points

**SCRUM**

Roles and events that constitutes a SCRUM

# Scenario based Medium question multiple unrelated questions

* Potential topic
  + Requirements
    - Acceptance criteria
  + Team dynamics

Given a scenario, how would team dynamics be formed/shaped or enhanced

# Scenario based Medium long question 1 Question

## Recommended to use bullte points

* Ethics and Licences
* SCRUM

Potential Questions

Discuss about a scenario to whether or not it is ethically acceptable

Discuss the appropriate license for an application. State your reasoning.

Discuss what sort of license you would use for your application. State reasoning.

# Scenario based medium question multiple related questions

# Ethics

* Pricacy
* Responsibility if done wrong?
* Done to the best of your abilities
* Quality testing
  + Even if you are implementing someone else’s work, don’t assume that they have done a good job at it
  + Did you do enough testing
    - Code reviews
    - Integration testing
* Look into policies before making any statements
* Software Liability
  + Responsible for injuries or defects in software applications.
  + Service vs product

# Code, Script

* Junit
  + Assert statements
  + Annotations
  + Edge cases
* Gradle
  + Build commands
  + Descriptions of the application of Gradle
  + Run
  + Gradle tasks
  + To list all of the tasks available
* Github, Git commands
* Jenkins

Diagram.

* Burn down Chart
* Velocity Chart

# Version control system

## What is Version Control System

In order to manage Artifacts in relation to time and preserve historic artifact objects. We use Version Control system to enforce perservation of work.

Version control system is a method of preservation of different version of artifacts in order to assist activities such as

* Revert back to previous version
* Compare changes between versions
* Search who was responsible for ecah changes

There are three main different methods of Version control systems

### Local Version control system

* + That stores the versions in a local file system
  + This is advantageous to individual works as it does not require any complication with server settings
  + However it becomes very difficult to distribute and collaborate ifmore than one person is in a group

### Central Version Control System

* + This method stores all the version information in a single server
  + It allows easy collaboration with consistent updates and eaily maintainable admin works
  + However it also is prone to system breakdowns as it is a single point of failure and if a server’s disk fails or server is down, the whole development team becomes bottlenecked

### Distributed VCS

* + This method allows distribution of git repository fully onto each collaborators
  + This means that there is no single point of location where each file is stored but it is stored under each Collaborators
  + It is advantageous because there is no single point of failures while following the advantages of Central VCS.
  + However, strict server management is required as all the repository is shared.

## What is GIT?

Git is a web based Distributed version control system that collects changes in content inside the repository and stores it as a blob in .git file.

IT allows atrifact preservation in a distributed VCS and easily management manner.

There are three main components of GIT workflow

* Modification of Files
* Staging of modified giles
* Commit staged files

Git tracks content that is set to be tracked, this is specified in .git file. These files are then tracked to see if there was any modifications made onto the files. If it does, git will collect **only the changes** made to those files and leave the unchanged files be. These changed files can then be **staged**, staging files processes the changed file into a blob file containing the changes made to the repo and links to the unchanged files. These are then placed to the staging areas which will then be **committed** to be stored perminantly under .git’s commit file.

## How does git collect changes and store them?

Git stores the files in a folder called .git which contains all the information about the past commits as well as any new files, modifications to tracked files and names for each commits etc.

When you initialise git or clone git, it is at a stable state. When changes are made (modification to files) or files are added/removed( untracked) git will record these changes and by doing git add, you will create a git blob object that will store the changes made into the repository. It is then stored in a staging folder, if the changes are not commited soon, it will be collected by git’s garbage collector and be deleted.

After which Git-commit should be called to perminantly store these changes into .git and convert the git changes into a stable git repository files.

## GitHub workflow

Git work is simple and very interactive to allow collaborations to be done easily

1. Create branch and checkout
   1. This is a step that is essential in differentiating dev’s working are with a stable application space so that no changes are made to the master branch while dev works on new implementation
2. Add commits
   1. This is a phase that allows dev team to make changes, implement new features and store them as a perminant changes under their personal branch
3. Open pull request
   1. This is a stage of collaboration with other team members to get feedback and code reviews on the changes and new implementations you made,
      1. Each team members are able to either approve, comment or request changes to the pull request
   2. As default, each team members would be able to traverse into eachother’s branches but this request explicitly asks them to go and have a look
4. Merge
   1. Merge the branch into the master branch to implement the new feature/ bug fixes etc.

## What is an Integration manager workflow

Integration manager is identified as the main manager who is able to merge to the master branch identified as the blessed repository. The hierarchy is broken down into three levels

* Integration manager
  + Who manages code reviews and process of merging the implementation into main master branch
* Team manager
  + Who manages the code reviews and maintain main branch for that specific teams. After completion of the team allocated work, it will request merge request to integration manager
* Dev team
  + Who does new implementations or bug fixes, they will pull request a team manager after completion.

# Gradle and Automatic application build

## What is configuration management?

It is a type of management that is responsible for tracking what compoenent versions incorportaed into each system versions, therefore managing the code-line and base-line for ecah system versions.

They are also repsponsible for managing changes in software system and analyse whether the changes are realistic.

It is broken down into 6 different parts

System building

* Assembling prohram component, data and libraries, then being able to compile all of these files to create an executablesystem

Version management

* Ensure changes made to each system components do not interfere with other component
* Keep track of artifact and system versions

Change management

* Keeping track of the changing requirement from client and estimate the impact of these changes

Release management

* Prepare application for external release and keeping track of system versioning

## What is Gradle

Gradle is a build automation tool that assists build process easier to maintain and automatically done.

Gradle’s main focus is to allow automation of

* Creating classes
* Manage build dependencies
* Construct executable files
* Run application
* Conduct testings as specified
* Construct test result report if needed

# Gradle life cycle is devided into three steps

1. Initialization which mainyl focuses on the identification of project to be participated in the build process
2. Configutation that manages assembling TASK objects into an internal object model called Directed Acyclic Graph that will then be utilised for building processs
3. Execution which is the step that build tasks in order specified in the configuration with the dependencies specified.

# Software Testing

## Define Software Quality Assurance

It is a process of ensuring that the software application satisfies the requirement specified by the client and to ensure that the application does not have any unwanted errors such as system crashes, incorrect computations, unnecessary interactions or data corruptions.

## Who does Software Testing?

Development team

* When they work on their own codes
* When they review eachother’s code

Independent Testers

* To allow fresh perspective and to ask for professional assistance for integrity

End users

* Who are able to find and identify any form of unwanted processes or errors for the development team to patch.

Or anyone who

* Has detailed understanding of the system
* Holds domain knowldege
* Knowldeged on testing techniques

## What are the different types of testing levels?

Unit/functional testing

* In order to verify functionality of Software Component independently

Integration Testing

* When we want to verify the interactions between software components

System Testing

* When we want to verify functionality and behaviour of the **entire software system**

Regression Testing

* When we want to continually test each integration to make sure absence of malicious interaction between new integrations

Acceptance testing

* When we want to verify client’s satisfaction of the application

## Types of testings

* Black box testing
  + When the decision of different testings are done without knowledge of the internal logics
* White box testing
  + When you decide your test cases dependant on the internal logic of software
* Defective testing
  + When you want to purposely break an application
* Validation testing
  + When you want to test to make sure that all the requirements are met

# What is Continuous Integration?

* Continuous integration is a practice in which the development team continually integrates their changes into the main workspace. This is aimed to ensure that the software is always on a updated state that is stable.
* This is also to minimise thee duration of integration testings as small changes would be easier to analyse if presence of an error

Continuous integration consists of different tools and practices working together

* Version Control system
  + Use of github to get updated each time a new integration is done onto the main workspace
* Automated build process
  + To be able to build and test the application with respect to confiuration mangement
* Workflow integrity and consistency
  + Team members to follow the workflows and conduct continuous integration
* Testing processes
  + To ensure that the latest commit onto the working space is stable.
* CI Server to encapsulate all the processes and automate them all

Continuous delivery and deployment

Both continuous delivery and deployment aims to produce software applications in short period of time to deliver quickly to the end users. This is essential as it will ensure continuous feedback and improve, change application accordingly.

However conitnous delivery puts the last state of deployment to be manual. This is advantageous if the dev team wants to have any form of last minute interaction prior to deployment

Continuous deployment however, focuses on automating all the processes including deployment stage. This allows the dev teams to not worry about manually deploying the application. But to know that if the build was done correctly. It will be deployed, or else they will be notifed for failed build

# Team dynamics

## How does Team dynamics play a role in Agile development

Team dynamics play a big role in Agile development team as it either improves or reduces the overall team performance and efficiency.

Due to the fact that most agile development team is small, each team members must be cross functional and responsible for each of their works while also respecting and collaborating with eachother with good communication skills.

However team dynamic factors such as

* Different personality
* Work style
* Roles
* Cultures

Have potential to interfere with team members being able to successfully collaborate with eachother

A good team would be able to have good team collaboration and cooportation skills and enhance the overall performance of the team.

## What would enhance team dynamics

If we were to refer to the Agile manifesto

* Build project around Motivated individuals, giv them the enviiornment and support hey need and trust them to get their job done
  + This would allow management of goals to be mutually focused allowing each team member to respect eachother’s dedication work together in a motivated manner to solve problems together

Moreover, you can conduct team building activities to encourage interpersonal relationsihps between team members to enahnce team dynamics

* Goal settings
  + Emphasise on construction of shared goals
* Interpersonal relationships
  + Provide good support to eachother
  + Communicate and share information
* Stand up meeting
  + Allowing face to face communication
* Roles and responsibility
  + Allows mutually agreed understandings of eachother’s roles that enhances the work space.

# SCRUM

## What are the main values of SCRUM?

* Commitment
* Courage
* Focus
* Openness
* Respect

Scrum is mainly for Teams who are small, cross functional, self-organising and focusses on delivering product iteratively.

## Requirements