- Explain what is meant by agile.

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end. It works by breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short two-week cycles called iterations.

- Incremental Development

An incremental process involves delivering components of software in parts. Each increment represents a complete subset of functionality and is fully coded and tested. For example, for an e-commerce store, initially payment can be supported only via credit and debit cards. In the next release, payment via wallets can be supported.al development

- iterative Development

An iterative process is one that makes progress through successive refinement. The development team develops first cut of the system, knowing in advance that some/many parts are incomplete. The team then iteratively enhances those parts until the product is satisfactory. With each iteration, customer's feedback is considered, and the software is improved through the addition of greater detail. For example, let us take an example of search functionality on a website. In for iteration, a simplistic search screen can be built. In next iteration, advanced search criteria can be added.

- Timeboxing

Under the Scrum framework, all activities are time boxed, also known as "timeboxing" or timeboxed, is to give a "fixed length" time segment to a specific event or activity. That unit of time is called a time box. The goal of timeboxing is to define and limit the amount of time dedicated to an activity. Scrum uses timeboxing for all of the Scrum events and as a tool for concretely defining open-ended or ambiguous tasks.

- Minimum viable product

An MVP is a version of the digital product which has just enough functionality to be tested and gather feedback about the product and its future development. It's one of the phases in full cycle product development. An MVP is 'functional' but it is not a fully-functioning product. It's an experiment – you may think you know what the results of that experiment will be (your hypothesis) but you build it and test it to be sure, so that the next stage of digital product design and development is based on a firm foundation.

Describe some of the online tools you can use to support agile development below and explain how they help the process

- version control systems

  Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter.

- build servers

  A build server, also called a continuous integration server (CI server), is a centralized, stable and reliable environment for building distributed development projects.

What are the roles of the Product Owner and Scrum Master in the SCRUM development approach?

- Product owner
  - Own the product vision
  - Represent the voice of the customer
  - Communicate with internal and external stakeholders
  - Align development work with the product roadmap
  - Write requirements and user stories
  - Organize the sprint
  - Refine the product backlog
  - Ensure acceptance criteria is met

- Scrum master
  - Coach the team in scrum practices
  - Facilitate scrum ceremonies (e.g., daily scrum meetings and sprint planning)
  - Help the team stay on task and complete sprint items
  - Identify and remove any productivity roadblocks

- Developers
  - Adhere to scrum practices (guided by the scrum master)
  - Determine how to complete the work prioritized by the product owner
  - Estimate how much work can be completed in a sprint
  - Communicate transparently about progress in daily scrum meetings
  - Complete all sprint items and deliver usable increments

- Describe the principles of TDD.

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only. This is opposed to software development that allows software to be added that isn't proven to meet requirements.

- Describe the TDD process.
    - Add a test: Write a test that defines a function or improvements of a function
    - Run test so it fails: This validates that the test harness is working correctly, that the new test does not mistakenly pass without requiring any new code
    - Write code and run test until it passes: The next step is to write some code that causes the test to pass. The new code written at this stage is not perfect
    - Refactor code: The growing code base must be cleaned up regularly during test-driven development.

- Describe two benefits of TDD.
    - TDD meant writing more tests and, in turn, programmers who wrote more tests tended to be more productive.
    - Large numbers of tests help to limit the number of defects in the code.
    - Forces code development to be modular with low coupling

- How can each module be tested in isolation?
    - Ensure low coupling
    - Use mocks to improve test isolation.

Agile development is seen as a way to manage risks in a project. For each of the following, explain how agile can reduce the risk

- Schedule slips.
  - Short release cycles
  - Burn-down charts / burn rate
- Business is misunderstood.
  - Customer is part of the team
  - User stories define business outcomes
  - Regular feedback by customer

- Business changes.
  - Minimum viable product
  - Iterative development delivers working code each time
- The project is suddenly cancelled!
  - Minimum viable product
  - Iterative development delivers working code each time
- The software is full of bugs
  - BDD / Acceptance testing defines automated suite to measure against customer requirements
  - TDD / Unit testing suite to automatically check for bugs in code modules
  - Regression testing keeps checking code that worked reviously.
  - Agile avoids false feature-rich which reduces the amount of code.

- Product Backlog
  - List of things that need to be done / captures system requirements
  - Anyone in the team can add to this list
  - List items should be user stories Can also include non-functional requirements such as improving performance or fixing bugs
  - The product owner prioritizes these after consultation with the rest of the team
  - Priorities are regularly adjusted based on current business needs

- Moscow Rules

  User stories fit into four different priority groups:

  - Must Have: the top items will be essential
  - Should Have: next there will be the important but lower priority tasks
  - Could Have: next are the low priority features that will be completed eventually Would Be Nice: the tasks at the bottom may never be implemented

- Product Sprint
  - A time-based unit of development / Restricted to a specific duration (timeboxed)
  - Duration of each sprint determined by team dynamics
  - Each sprint results in the delivery of working code

- Many teams use a Kanban board. Evaluate how these can be used to improve productivity
  - Keeps track of tasks in a visual manner
  - Tasks assigned to specific developer forces ownership
  - Each member of team knows what they need to achieve
  - Easy to identify issues early and resolve
  - Live document (information radiator)

- Compare the process of a daily stand-up to traditional meeting styles
  - Entire team must be present
  - Short time-box
  - Each member must contribute
  - Clearly defined set of questions/agenda

Modern agile teams often integrate Continuous Integration and Continuous Deployment into their workflow.

- Compare Continuous Integration with alternative approaches
  - All code goes into a central repository which auto builds test environment / test environment is built locally on dev machines.
  - All builds are automated / automated tests are triggered manually.
  - All code is built and tested every time code is pushed / tests are run before code is committed and pushed.
  - Integration takes place every push / Manual process.
  - Any errors are reported / All errors are captured by tests run locally

- Compare and contrast Continuous Deployment with alternative approaches
  - The acceptance tests define the product requirements / there may be other requirements not defined in the test suite.
  - If all tests pass the user story has been implemented / decision is taken by dev team.
  - Next step is to deploy to the live server automatically / process triggered by dev team manually.
  - This should be triggered automatically if tests pass / decision taken by dev team based on other factors.

- Appraise three challenges faced by organizations when adopting CI and CD
  - Needs team to fully adopt TDD/BDD
  - Organizational: need to agree control and workflow
  - Process: sign-off processes can be convoluted and time consuming
  - Technical: requires expertise in many different tools and techniques

At the core of agile development is the ‘Agile Manifesto’. Appraise the following:

- Individuals and interactions over processes and tools.
  - This first value places its emphasis on teamwork and communication.
  - Teams of people build software systems, not tools.
  - And to do that they need to work together effectively through productive interactions.

- Working software over comprehensive documentation.
  - Take the time to develop software that is clear, self-explanatory, and caters to the tasks that users need to get done

- Customer collaboration over contract negotiation.
  - Only your customers can tell you what they want, and it’s your job to listen. Successful development teams work closely with their customers and communicate with them frequently.

- Responding to change over following a plan
  - Change is the reality of software development – technology changes, business trends change, customers change. There is nothing wrong with a project plan – however, it must be malleable.
  - There must be room to allow for change and to respond to it otherwise your plan quickly becomes obsolete.