Sure! The image you shared is a **mind map of ISO 9126** – a standard that defines **software quality attributes**. These are the important characteristics that help us evaluate how good a software system is.

Let's go through **each quality attribute** (highlighted ones too) with **simple explanations and examples** 👇

✅ 1. Functionality

How well the software does what it's supposed to do.

- **Suitability**
  Is the software suitable for the purpose?
  - *Example:* A calculator app includes basic functions like addition, subtraction, etc.
- **Accuracy**
  Does it give correct results?
  - *Example:* A weather app correctly shows temperature and rainfall predictions.
- **Interoperability**
  Can it work with other systems or software?
  - *Example:* Google Docs allows importing from Microsoft Word.
- **Compliance**
  Does it follow legal and industry rules?
  - *Example:* A hospital's system follows health data privacy laws (like HIPAA).
- **Security**
  Is the system safe from unauthorized access?
  - *Example:* A banking app requires fingerprint login or OTP.

✅ 2. Reliability

How dependable is the software under various conditions?

- **Maturity**
  How stable is the software over time?
  - *Example:* An app that's been around for years with rare bugs (like Gmail).
- **Recoverability**
  Can the system recover from failure?
  - *Example:* A game that crashes but restores the last saved level when reopened.
- **Fault Tolerance**
  Can it still work if something goes wrong?

    - *Example:* A payment app continues working even if one of the servers fails.

✅ 3. Usability

How easy it is to use and learn the software.

- **Learnability**
  How quickly can a new user learn it?
    - *Example:* Canva is easy for beginners to learn graphic design.
- **Understandability**
  Can users understand how it works?
    - *Example:* A dashboard with clear icons and tooltips for each button.
- **Operability**
  Is it easy to operate and control?
    - *Example:* A music app with simple play, pause, and skip buttons.

✅ 4. Efficiency

How well the software uses system resources and responds.

- **Time Behaviour**
  Does it respond quickly?
    - *Example:* Google search results appear in less than a second.
- **Resource Behaviour**
  Does it use memory, CPU, battery efficiently?
    - *Example:* A note-taking app that runs smoothly without draining battery.

✅ 5. Maintainability

How easy it is to modify, fix or improve the software.

- **Stability**
  How often does it crash when changes are made?
    - *Example:* Updating one module in a shopping app doesn't break the checkout.
- **Analysability**
  How easily can you understand and analyze the code?
    - *Example:* Well-commented source code makes bug-fixing easier.
- **Changeability**
  How easily can you make changes?

    - *Example:* Adding a new feature to a modular app is easy.
- **Testability**
  Can you test it easily?
    - *Example:* A codebase with unit tests is easier to test after updates.

✅ 6. Portability

How easily the software can run on different systems or environments.

- **Installability**
  How easily can it be installed?
    - *Example:* Zoom installs in a few clicks on Windows or macOS.
- **Conformance**
  Does it conform to portability standards?
    - *Example:* A Java app runs on any OS with Java installed (thanks to JVM).
- **Replaceability**
  Can one system replace another without issues?
    - *Example:* Switching from one PDF reader to another without losing settings.
- **Adaptability**
  Can it adjust to different environments?
    - *Example:* A website that adjusts layout for both phones and desktops (responsive design).

Let me know if you'd like a downloadable table or a PDF version of this summary!