🌟 Main Goal of SQE:

"To meet or exceed quality expectations through appropriate QA activities, while minimizing cost and project risks within project constraints."

📌 What does this mean?

Imagine you're building an app. People expect it to run fast, not crash, and be easy to use. The goal of Software Quality Engineering is to **make sure that happens** using good **Quality Assurance (QA)** practices — like testing — **without spending too much money or going over deadline**.

🧩 SQE is Part of the Software Engineering Process

- It's **not separate** — it works along with other concerns like:
  - ⏰ **Schedule**
  - 💰 **Cost**

Example: If you need to finish the app in 3 months and under $5,000, your QA plan must also respect that timeline and budget.

🔁 SQE Activities – Generic Testing Process

Systematic testing based on formal models

This just means we **follow a clear, step-by-step approach** using some predefined models or rules, not just testing randomly.

🛠 Pre-QA Activities: Quality Planning / Test Planning

✅ Most of the key testing decisions are made here.

This is like **planning before building a house**. You decide:

- What quality means for your app,
- What kind of tests you'll run,
- And how much you're willing to spend.

✅ Breakdown of Pre-QA Activities:

❖ Set Specific Quality Goals

(High-level activities to test planning)

You're setting your **destination**. What does “quality” mean for your app?

Example goals:

- The app should crash less than once per week.
- It should load within 2 seconds.
- Users should rate it 4 stars or more.

❖ Identify Quality Perspective and Expectation

What do **users** or **clients** care about?

If you're making a mobile game:

- Players may care about **fun** and **performance**. If it's a banking app:
- Users care about **security**, **accuracy**, and **uptime**.

So, quality depends on the **type of users and their needs**.

❖ Select Direct Quality Measures

These are **numerical targets** for your quality goals.

Example:

- **Efficiency:** App loads in ≤ 2 seconds.
- **Reliability:** Uptime is 99.9%.
- **Usefulness:** 90% of users complete the signup process.

We measure these to check if we’re on track.

❖ Assess Quality Expectations vs. Cost

How much does it cost to meet certain quality levels?

Example:

- Getting 99% uptime might cost $1000.
- But getting 99.999% uptime might cost $10,000.

We balance **how good** we want the app to be with **how much** we can afford.

🧠 Form an Overall QA Strategy

(Low-level activities like test case creation and test preparation)

Now that we know our goals, we plan how to test the app **in detail**.

✅ Select Appropriate QA Activities

What types of testing do we need?

Examples:

- **Unit Testing:** For each small part of code.
- **Integration Testing:** Making sure parts work together.
- **System Testing:** Testing the whole app.
- **User Acceptance Testing (UAT):** Checking with real users.

✅ Choose Appropriate Quality Measurements and Models

Tools to **track and improve** quality.

Example:

- Use bug-tracking tools to see how many bugs remain.
- Use performance monitoring tools to track speed/load time.

🧪 Test Procedure Preparation

✅ Preparing Test Cases (micro-level)

Now we get into actual **test case creation** — this is like writing down **step-by-step instructions** to test a feature.

📄 Test Case Definition

A test case is a **set of inputs, conditions, and expected results** for a specific feature.

Example:

- **Test Case for Login Feature:**
    - Input: Correct username and password
    - Expected result: User is taken to the dashboard
    - Input: Wrong password
    - Expected result: “Incorrect Password” error

📦 Test Case Allocation

Assigning different test cases to **different testers** or **testing stages**.

Example:

- Tester A checks login.
- Tester B checks payment.

🔄 Sequencing of Individual Test Cases

Running tests from **simple to complex**.

Why?

- It helps catch basic issues first, so you don’t waste time on big complex features that might break because of small bugs.

💡 Summary in Simple Terms

Step	**What You Do**	**Example**
1. Plan	Set quality goals, user expectations, and decide what to test	Make sure your banking app is secure and fast
2. Measure	Choose what numbers to track (speed, reliability)	Target: 2-second load time
3. Strategy	Pick what kind of testing to do and how	Use unit + system + UAT tests
4. Test Cases	Create test scripts with inputs/outputs	Test login with correct/incorrect info
5. Execute	Run the tests in order	Start with login, then payments, etc.

📝 TEST PLAN

✅ What is a Test Plan?

A **test plan** is a **high-level document** that outlines **what, how, and when** testing will happen.

📌 Key Points:

- **Objectives** → What do we want to achieve?
  E.g., Find all critical bugs before product release.
- **Scope** → What parts of the software will be tested?
  E.g., Only frontend login and dashboard modules in Phase 1.
- **Approach** → What type of testing? Manual, automated, black-box, etc.
- **Resources** → People, tools, machines involved in testing.
- **Schedule** → Timeline of when testing will be done.
- **Focus** → What are the main areas of concern?
  E.g., Focus on security and performance testing.

📘 Example of a Simple Test Plan:

Section	Details
Objective	Ensure payment system works correctly
Scope	Test credit card and PayPal payment
Approach	Manual testing + automated script for PayPal
Resources	2 QA engineers, 1 test server
Schedule	April 20 - April 25
Focus	Functional bugs + edge case validation

🧪 TEST CASE

✅ What is a Test Case?

A **test case** is a **low-level document** that gives **step-by-step instructions** to test a specific feature.

📌 **What does it include?**

- **Input** → What are you giving to the system?
- **Action/Event** → What will the user/system do?
- **Expected Result** → What should happen if it works correctly?
- **Execution Conditions** → Any setup needed before testing.

📘 **Example Test Case:**

Field	**Value**
Test Case ID	TC001
Input	Email: user@test.com, Password: 123456
Action	Click "Login"
Expected Result	User is taken to Dashboard page
Pre-condition	User account must already exist

⚠️ **Note:**
Test Plan = Big picture strategy.
Test Case = Tiny steps to check specific things.

📦 **TEST SUITE (Macro-Level)**

✅ **What is a Test Suite?**

A **test suite** is a **collection of related test cases** that you run together.

Think of it like a **playlist** of test cases — they run in sequence.

📌 **Key Points:**

- Test cases in a suite should be:
  - Organized in **sequence** (usually from simple to complex)
  - Based on **specific testing techniques**
  - Sometimes **reused** from older versions — this is called **regression testing**

📘 Example Test Suite (for Login Feature):

1. Test Case 1: Valid email and password → Success
2. Test Case 2: Invalid password → Show error
3. Test Case 3: Empty fields → Show validation
4. Test Case 4: SQL injection attempt → Block access

⚙️ In-QA Activities: Test Execution

You're now **running** the tests you've planned.

📌 Key Activities:

- ✅ **Execute Test Cases**
- 🐞 **Handle Discovered Defects**
    - Log details like *what happened, where, when, and how bad it is* (severity)
- 📝 **Document Everything**
    - Useful for future testing and audits
- 📊 **Measure with Templates**
    - Use a form to record pass/fail results, defect count, time taken, etc.

📈 Post-QA Activities: Measurement, Assessment & Improvement

These are the **follow-up tasks** after tests start running.

📌 Key Points:

- 🧠 **Analyze Results**
    - Look at how the app performed — were goals met?
- 🔧 **Improve Process**
    - Fix recurring issues in test process or development
- 🔄 **Run in Parallel**
    - These activities don't only happen after testing ends. They **run alongside** QA too.

- 🧩 **Overlap**
  - Pre-QA, In-QA, and Post-QA can happen at the same time.

👨‍👩‍👧‍👦 TESTING TEAMS: Organization & Management

✅ Who does testing?

- 👥 **Customers/Users**: Can help test usability (called **beta testing**)
- 🏢 **Independent Testing Organizations**: External experts hired to test
- 🧑‍💻 **In-house QA Teams**: Part of the software company itself

---

📌 Test Team Models:

1. **Vertical Model**
   - Dedicated people test **one product** from top to bottom.
   - *Example: Team A only works on your Android app.*
2. **Horizontal Model**
   - One team does **one kind of test** (like security testing) for **many products**.
   - *Example: Security testing team handles Android app, iOS app, and website.*
3. **Mixed Model**
   - Combines vertical + horizontal.
   - *Used in large companies.*
   - *Example: A team works deeply on the product, but gets help from a separate performance testing team.*