P P SAVANI UNIVERSITY

TUTORIAL NO.:8
ON
SOFTWARE ENGINEERING(SSCS3010)

TITLE: Verifying Requirements Against Quality Attributes

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY (BSC-IT)

SUBMITTED TO:	**SUBMITTED BY:**
Name: MS. HEMANGINI MEHTA(HGM)	**Name: RAJ MO FAHIM ZAKIR**
Designation: ASSISTANT PROFESSOR	**Enrollment: 23SS02IT161**
P P Savani University	**BSCIT5B-Batch 2023-26**

Faculty Signature: ____________

INSTITUTE OF COMPUTER SCIENCE AND APPLICATIONS
P P SAVANI UNIVERSITY
MANGROL, SURAT- 394125 (GUJARAT)

TUTORIAL-8 **Date:29/08/2025**

Aim: Verifying Requirements Against Quality Attributes

1. What are Requirements?

- **Functional Requirements** → what the system should do (e.g., "The system must allow users to log in with an email and password").
- **Non-Functional Requirements (Quality Attributes)** → how the system should behave (e.g., "The system must respond within 2 seconds").

2. What are Quality Attributes?

Quality attributes describe **system properties** that affect user satisfaction and system performance.
Examples:

- **Performance** (response time, throughput)
- **Reliability** (fault tolerance, availability)
- **Security** (authentication, data protection)
- **Usability** (ease of use, accessibility)
- **Scalability** (ability to handle growth)
- **Maintainability** (ease of modification)

3. Why Verify Requirements Against Quality Attributes?

- To ensure that requirements are **measurable and testable.**
- To check **consistency** (no conflicts between requirements).
- To confirm that the system will **meet user expectations** in real conditions.

✅ Steps to Verify Requirements Against Quality Attributes

Step 1: Identify the Quality Attributes Relevant to Your System

Example: For an **Online Food Ordering System**, the important quality attributes may be **Performance, Security, and Usability**.

Step 2: Map Requirements to Quality Attributes

Create a **Requirements vs Quality Attributes Matrix**.

Requirement	Performance	Security	Usability

R1: Users must log in with email & password	Low	High	Medium
R2: System must display restaurants within 2 seconds	High	Low	High
R3: Online payment must be encrypted	Medium	High	Low
R4: Mobile app must support English & Hindi	Low	Low	High

Step 3: Define Verification Criteria

Each requirement should be **testable** against quality attributes.

Example:

- **Performance**: “System should display available restaurants in ≤ 2 seconds under 1000 concurrent users.”
- **Security**: “All payment data must be encrypted using AES-256.”
- **Usability**: “A new user should be able to place an order in ≤ 5 minutes without training.”

Step 4: Verification Techniques

- **Reviews & Inspections**: Check if requirements are **clear, consistent, measurable**.
- **Prototyping**: Validate usability and performance before implementation.
- **Modeling & Simulation**: Test system behavior under load.
- **Testing Plans**: Ensure that non-functional test cases exist (load testing, security testing).

📌 Example Walkthrough

Case: Online Food Ordering System

- **Requirement**: “System must allow secure online payment.”
- **Quality Attribute**: **Security**
- **Verification**:
  - Check requirement for **clarity**: What kind of payment security?
  - Add details: “System must use HTTPS protocol and PCI-DSS compliance for all transactions.”

- **Testability**: Security testing team validates encryption protocols & compliance.

Requirements vs Quality Attributes (Resume Builder for Freshers)

1. Identify Requirements

Functional Requirements (FRs):

1. User can register and log in using email and password.
2. User can create and edit a resume with personal details, education, skills, and projects.
3. User can upload a profile picture and documents.
4. User can generate/download resume in PDF format.
5. User can choose from multiple resume templates.

Non-Functional Requirements (NFRs / Quality Attributes):

- **Performance:** Resume should generate quickly.
- **Security:** Protect user data & resumes.
- **Usability:** Easy for non-technical users.
- **Scalability:** Handle many users at once.
- **Maintainability:** Easy to update templates/features.

2. Requirements vs Quality Attributes Matrix

Requirement	Performance	Security	Usability	Scalability	Maintainability
User Registration & Login	Response ≤ 2s	Encrypted passwords (bcrypt/argon2), HTTPS	Simple signup flow	Supports 10k+ users	Easy to update login module

Create & Edit Resume	Save in ≤ 1s	Validate input fields	Simple form UI	Handle 500 concurrent edits	Modular form fields
Upload Profile Picture/Documents	Upload ≤ 3s	Virus scan, size limit, HTTPS upload	Drag & drop support	Handle bulk uploads	Pluggable file storage service
Generate/Download Resume (PDF)	Generate in ≤ 5s	Secure PDF link (time-limited)	One-click download	1000+ resumes/hour	Template engine support
Select Resume Template	Load in ≤ 2s	No injection in template code	Preview before apply	Add new templates without downtime	Config-driven template updates

3. Verification Criteria (Examples)

- **Performance:** Resume PDF must be generated within 5 seconds for up to 1000 concurrent users.
- **Security:** All user passwords encrypted with bcrypt; resume download links expire in 10 minutes.
- **Usability:** A first-time user should be able to build a resume in ≤ 10 minutes without guidance.
- **Scalability:** System should support 10,000+ registered users and 1000 concurrent sessions.

- **Maintainability:** Adding a new resume template should not require code changes—only config updates.

4. Verification Techniques

- **Reviews & Inspections:** Ensure requirements are specific and testable.
- **Prototyping:** Test resume creation UI with sample users.
- **Load Testing:** Simulate multiple users generating resumes.
- **Security Testing:** Check encryption, secure uploads, expired links.
- **Regression Testing:** Ensure new templates don’t break existing ones.