Requirements Engineering

It is the process of identifying, eliciting, analyzing, specifying, validating and managing the needs & expectations of stakeholders for a software system.

Requirement Engineering Process

→ Feasibility study.
→ Requirement elicitation
→ " specification.
→ " verification & validation.
→ " management.

Feasibility Study

Technical Feasibility: It assesses available hardware, software, technology and team capabilities to determine project visibility.

Operational Feasibility: It evaluates how well the system meets requirements, its ease of use and maintenance and the acceptability of the proposed solution.

Economic Feasibility: It assesses project costs, including development, resources & operations, to determine financial viability and benefits.

Legal Feasibility: It ensures the project compiles with laws, regulations, contracts and intellectual property requirements to avoid legal constraints.

Schedule Feasibility: It evaluates project timeline, milestones, resource availability and time constraints to ensure timely completion.

Requirements Eliciation

It gathers stakeholder needs and domain knowledge using various techniques to define software system requirements.

Techniques:

Interviews: These are one on one conversations with stakeholders to gather information about their needs & expectations.

Surveys: These are questionnaires that are distributed to stakeholders to gather information about their needs & expectations.

Focus Groups: These are small groups of stakeholders who are brought together to discuss their needs & expectations for the

Observation: This technique involves observing the stakeholders in their work environment to gather information about their needs and expectations.

Prototyping: This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

Requirement Specification

It is formally documents functional & non-functional requirements, constraints and models [ERD, DFDs, FDDs (Function decomposition diagrams)] to ensure clarity, prioritization, and completeness for developers & stakeholders.

Steps:

Functional Requirements: It defines the specific operations, features, and behaviours a software system must perform.

Non-functional requirements: It defines the system's quality attributes, including performance, reliability, usability & security.

Constraints: These describe any limitations or restrictions that must be considered when developing the software system.

Acceptance Criteria: These discribes the conditions that must be met for the software system to be considered complete & ready for release.

Requirements should be clear, simple and consistent, using natural language, visual aids and reviews to ensure completeness & accuracy.

P.T.O

07.9

Requirements Verification & validation.

Verification: It refers to the set of tasks that ensures that the software correctly implements a specific function.

Validation: It ensures the software aligns with customer requirements, preventing errors from propagating to later stages and reducing rework. It includes:

→ The requirements should be consistent with all other requirements; no two requirements should conflict with each other.

→ The requirements should be complete in every sense.

→ The requirements should be practically achievable.

P.T.O

Other necessary points on V&V:

→ Verification ensures requirements are complete, consistent, clear, testable & error-free through reviews, models & stakeholder meetings.

→ Validation ensures requirements meet stakeholder needs & expectations through testing, simulations & prototypes.

→ V & V are iterative processes involving stakeholders & the development team to ensure through review and testing of requirements throughout (SDLC).

Requirements Management

It involves analyzing, documenting, tracking, prioritizing & controlling changes to the SRS, ensuring requirements remain valid, relevent, and aligned with stakeholder needs throughout the SDLC.

Several Key activities:

Tracking & controlling changes: It involves monitoring, assessing impacts & approving or rejecting changes to requirements during development.

Version Control: This involves keeping track of different versions of the requirements document and other related artifacts.

Traceability: This involves linking the requirements to other elements of the development processes, such as design, testing & validation.

Communication: This envolves ensuring that the requirements are communicated effectively to all stakeholders and that any changes or issues are addressed promptly.

P.T.O

Monitoring & Reporting: This envolves monitoring the progress of the development process & reporting on the status of the requirements.

Tools Involved in RE (Requirement Eng)

- → Observation report
- → Questionnaire
- → Usecases
- → User stories
- → Requirement Workshop
- → Mind mapping
- → Role playing
- → Prototyping

Advantages of RE

- → Ensure stakeholder needs are met.
- → Identifies issues early.
- → Enhances cost-effectiveness.
- → Improves communications.
- → Clarifies requirements.
- → Resolves conflicts.
- → Ensure timely, quality delivery.
- → Reduces risks.
- → Strengthens development foundation.

Disadvantages of RE

→ Time-consuming & costly.

→ Hard to meet all stakeholder needs.

→ Ensuring clarity & consistency is tough.

→ Changes cause delays & extra costs.

→ Complex requirements increase effort.

→ stakeholder conflicts are challenging.

→ Difficult to align stakeholders understanding.

Security

Properties of Secure Softwares

→ Confidentiality
→ Integrity
→ Authentication.
→ Non-repudiation.
→ Availability.

Confidentiality

It ensures that sensitive data is accessible only to authorized parties, preventing unauthorized access & leaks while protecting privacy.

Example:

→ GPA should only be visible to the particular student involved.

P.T.O

Integrity

It ensures that data is only modified by authorized parties & remains protected from unauthorized alterations or damage.

Example:
→ Submissions are not edited by anyone other than student.

Authentication

It verifies the identity of an individual or entity, ensuring that requests to a system comes from legitimate sources.

Example:
→ should be able to determine if the user is indeed a student or an instructor.

Non-repudiation/Accountability

It ensure that actions can't be denied by their originator, balancing accountability with privacy.

Example:

→ students can deny to have edited submission after the deadline.

Availability

It ensures that a system remains accessible & responsive users at all times, preventing disruptions like Denial of service attack.

Popular cyber attacks

→ Malware
→ password Attack.
→ SQL Injection.
→ Zero Day Attack.

Malware

→ Unwanted software that is installed on your system without your consent.

→ Ransomware: This is malicious software that holds your data hostage until a ransom is paid.

→ Spyware: These are programs installed to collect confidential information about users.

→ Trojan horse: A seemingly legitimate program, but with a malicious intent.

→ Logic Bomb: This type of virus is capable of being triggered at a precise moment.

Password Attack

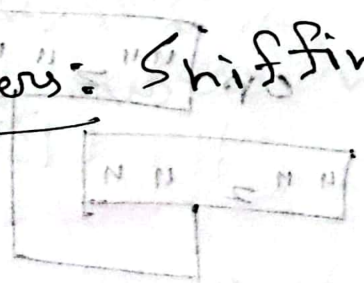
Brute Force/Dictionary Attack: Try by using common passwords.

Phishing: Sending fraudlant but attractive emails

Man-in-the-middle: Sniffing the communication packets.

Credential stuffing: Signed in from another device.

Keyloggers: Sniffing while typing the password.



Social Engineering: Manipulating into giving up the password.

SQL Injection

It is a web hacking technique, where malicious SQL code is inserted via user input, potentially compromising or destroying the database.

Example:

→ Select * from user where

id = " " or 1 = 1

→ Always true

→ where name = " " or " " = " " AND

password = " " or " " = " "

→ Always true

→ valid SQL & return all rows from the user table.

SQL Injection Prevention

It can be prevented using SQL parameters which ensure user input is treated as data rather than executable code by SQL engine.

→ Where id = @0

@ markers

Zero Day Attack

It exploits unknown software vulnerabilities before developers can patch them, often targeting users who delay updates.

Good Practices

→ Fields length checking.

→ Validate input on both client & server side.

P.T.O

→ Use high-level languages (Java, C#) to reduce exploits.

→ low-level languages (C, C++) are vulnerable to buffer overflow.

→ Deploy apps only in authentic stores.

→ Use public key Cryptography (AES, DES) for encryption.

→ Enforce strong passwords & periodic changes.

→ Ensure authentication over encrypted channels (HTTPS)

→ Do security testing.

Documentation

It is essential for ensuring usability & maintainability of a software.

Types of Software Doc

→ Requirement Specification (RS)
→ Architecture/Design
→ Technical
→ User End
→ Marketing.

Types of RS

→ BRS
→ FRS
→ SRS

Business Requirements Specification (BRS):

→ Formal document outlining customer business needs.
→ Developed by business analysts based on stakeholder input.

P.T.O

→ Created at the start of the product life cycle to define core goal.
→ High-level document detailing all client requirements comprehensively.

Functional requirements Specifications (FRS):

→ Describes all functions the software must perform.
→ Outlines step-by-step operations from start to end.
→ Details software behavior during user interaction.
→ Developed through collaboration between testers & developers.

SRS:

→ prepared by system analysts to describe software requirements.
→ Include functional & Non-functional requirements with use cases.

→ Serves as an agreement with stakeholders on product functionality.

→ Acts as a foundational framework for the project.

→ Ensure alignment among team-members accross departments.

Architecture Doc

Comprehensive system overview with multiple architectural views. It includes:

→ ERD, UML, DFD.

Technical Doc

→ Covers technical Aspects of the project.

→ Include descriptions aiding development.

→ Contains info on libraries, dependencies, APIs & code base.

P.T.O

User-End Documentation

Guides end users on using a product or service. It may include:

→ Snapshots.
→ Diagrams
→ Tutorials
→ Steps
→ Do's / Don'ts

Marketing PoC

Promotional metarials designed to attract potential customers. It requires:

→ Exploit human psycology.
→ Study current trends.
→ Clearly describes product features.
→ logical comparison with the alternatives.

Fall - 2023

2(a)

More beneficial for the current and potential users of a software:

→ User-End Documentation.

→ Marketing Documentation.

User-End Doc:

→ Guides end users on using a product or service.

→ Snapshots, Diagrams, Tutorial, steps, Do's/Don'ts.

Marketing Doc

→ Exploit human psycology.

→ Study current trends.

→ describes product features.

→ logical comparison with the alternatives.

3(a)

Requirement Verification:

It refers to set of tasks that ensures that the software correctly implements a specific function.

Validation:

It ensures the software align with the customer requirements, preventing errors from propogating to later stages and reducing rework.

Example:

Verification: "The system shall display an error message if the password is not correct"

Validation: Asking the user: "Do you need an error message when login fails?"

3(b)

Confidentiality

→ Sakib accessed Tamims private chats with papon.

→ Unauthorized access to private info.

Integrity

→ Sakib posted as Tamim & changed his username.

→ Data was modified without permission.

Authentication

→ Sakib acted as Tamim.

→ System fails to verify true identity.

P.T.O

Summer 2023

2(a)

BRS (Business Requirements Specifications)

→ Formal documents outling customer business needs.

→ Developed by business analysts based on stakeholder input.

→ Created at the start of the product life cycle to define core goal.

FRS (Functional RS)

→ Described all the function the software must perform.

→ Outlines step-by-step operations from start to end.

→ Developed through collaboration between testers & developers.

2(a)

Requirement Elicitation

It gathers stakeholder needs & domain knowledge using various techniques to define software system requirements.

[Interviews, Surveys, Focus group, observation, Prototyping]

Requirement Analysis:

Clarifying, organizing & checking these requirements for conflicts, completeness, consistency, feasibility.

2(b)

Authentication

→ The system allows access, when they enter another user's password.

→ failure in varifying correct user's identity.

P.T.O

Confidentiality

→ User can view & alter data of other random users.

→ Unauthorized access to sensitive data.

Integrity

→ User can alter another users data.

→ Data was modified without permission.

Spring-2024

2(a)

Functional Requirement: operations, features, behaviours a software must perform.

→ reset password via email.

Non-Functional Requirement

→ performance, reliability, usability & security.

→ handle 1000 users without performance degradation.

3(a)

User-end documentation

→ Helps user how to use the AR glasses.
→ Uses pictorial symbols. [Face, glasses]
→ Ideal for manuals, guides or quick-start instructions.

I Prefer Doc-B over Doc-A:

→ Show cause-effect clearly.
→ ~~[illegible]~~ Combines visual to reduce confusion.
→ Avoids ambiguity seen in Doc-A's seperated icons.

3(b)

Case-1

SQL Injection.

→ An attacker uses SQL injection to input malicious code, bypassing security & accessing sensitive data.

Process

→ Attacker input malicious code.

→ System sends it to database without proper check.

→ SQL query reveals restricted data.

Case-2

Password Attack

→ By keyloggers Attacker gather all the password of users account.

→ Then this rouge recipe has been shared by attackers from all the accounts.

Sum-24

Fall-24

Summer - 24

2(b)

Auxthwo (Malware)

→ Spyware → Free wifi

→ computer asking him to enter the username & password for one of his social media account.

→ By using free wifi, it installed programs to collect confidential information about users.

John (Password Attack)

→ keyloggers

→ Someone gain access to these accounts as soon as he enters his username & password [shiffing while typing the password.

3(a)

1st phrase → User-End Doc

2nd phrase → Marketing Doc.

3rd phrase ↗

Fall-24

2(b)

→ log into any account [Confidentiality, Authentication]

→ find editable text, which you can change easily [Integrity]

→ freezes for a few hours [Availability]

P.T.O

3(a)

Architectural Doc

→ Comprehensive system overview with multiple architectural views.

→ ERD, UML, DFD

Technical Doc

→ Cover technical Aspects of the project.

→ Include descriptions aiding development

→ Libraries, dependencies, APIs, code base

→ how to set-up development environment.

P.T.O

Github

Fall - 2024

1(a)

(I) **Download Repo**

- $ git clone www.github.com/Coolrepo

(II) **Create 2 new branches**

- $ git branch Audio
- $ git branch Video

(III) **Create 2 new files**

- created "song.txt" & "Movie.txt"

(IV) **Edit song.txt & commit in Audio Branch**

- Edited "song.txt"
- $ git checkout Audio
- $ git add song.txt
- $ git commit -m "Edited song.txt in Audio Branch"

(v) Update "Movie.txt" & Commit in Video Branch

- updated "Movie.txt"
- $ git checkout video
- $ git add Movie.txt
- $ git commit -m "updated movie.txt in Video Branch"

Create Partj.txt & commit in Audio Branch

(vi) - Created "Partj.txt"
- $ git checkout Audio
- $ git add Partj.txt
- $ git commit -m "created Partj.txt in Audio Branch"

(vii) Revert back to previous commit in Audio B

- $ git log
- $ git reset --hard abc123 → hashcode.

(viii) Merge all Branches to main & upload to git

- $ git checkout master
- $ git merge Audio
- $ git merge Video.
- $ git push.

Summer - 2024

1(b)

(i) Download repo

- $ git clone www.github.com/FunRepe.

(ii) Create 2 new files

(iii) Create 2 new branches

(iv) Edit & commit

(v) Edit & commit

(vi) Merge your work with the recent version in github.

- $ git pull origin main

(vii) Delete & commit

- Deleted "Jerry.txt"
- $ git checkout CNJr
- $ git add Jerry.txt
- $ git commit -m " deleted & commit "

(viii) Merge & upload.

Spring - 2024

1

1. **Initialize a Git Repo**

- $ git init

2. **Set your username & email**

- $ git config --global user.name "Sadman"
- $ git config --global user.email "ww@gmail.com"

8. **Show history of the file txt.txt**

- $ git log --txt.txt

Summer - 2023

4(a)

Show the differences between 1st & 2nd commit of "b-2"

- $ git diff a11111 a22222

Findout all commits "f2.txt"

- $ git log -- f2.txt

Delete the branch "b2"

- $ git branch -d b-2

Fall-2023

Checkout to the commit with hash d7b7953. Now, delete all commits made after this

- $ git checkout d7b7953
- $ git reset --hard d7b7953