INDEX

S.No.	Date	Name of the Experiment / Programme / Practical	Page No.	Remarks
1	26/8/24	Software engineering principles	1	
2	9/9/24	planning and development planning organisation structure	7	
3	23/9/24	Software Cost factors & estimation techniques	14	
4.	26/9/24	S.R.S & Formal specification system.	18	
5.	29/9/24	Design notations & Design Techniques.	19	
6.	30/9/24	realtime distribution systems & Test plan.	26	
7.	4/10/24	user interface design	30	
8.	8/10/24	Human interface design	33	
9.	18/10/24	Software quality Assurance.	36	
10.	20/10/24	Black Box Testing		

Date: 26/8/24 Page No. 1

Assignment No.: 1

Assignment Topic: Software Engineering principles.

1. Explain about Software Engineering principles:-

A. Software Engineering principles:-

1. Abstraction - focusing on factors and hiding complexity.
2. modularity - Breakdown the system into smaller that is independent
3. reusability - Designing the software to be reusable in other context
4. scalability - Building the software to development
5. reliability - Encouraging the software works.
6. security - protecting the software & Data.

Software Engineering Benifits:-

there are few benifits or advantages of SE.

1. Improving quality
2. Increase productivity.
3. Better Management.
4. Enhanced customer satisfaction.

1. Improving Quality:-

In SE improving the quality of software which should have more reliability, Efficient & maintainance of the software then only software will be improved as a quality factor.

2. Increase productivity:-
To increase the productivity in the Software Development process and reduce the errors which is occured in the software, so that productivity of software can be developed.

3. Better management:-
In this software management should think more efficient planning & coding of the co-ordination of the team members and control of the Software.

4. Enhanced customer satisfaction:-
The software that is used by the user & developed than the customer will be satisfied then only the software as giving proper that result will satisfy the customer satisfaction.

Software Engineering Disadvantages:-
In SE there are few challanges can be accepted by the software.

1. Complexity
2. Change management
3. team collaboration.
4. time & Budget to the Software.

1- Complexity:-
In the software we have to manage interact with the software system then only software can be work as properly.

2. Change management:-
In the software adapting to changing the

requirement and changing the various technology like planning and maintaining of the software.

3. Team collaboration :-

when we are developing a software Co-ordinating various devices and various team members to develop the software..

4. Time & Budget to the Software :-

In this Software meetings and dead lines of the software. at the same time we have to think software development budget limitation that is cost of the software.

. Conclusion :-

In this SE principles and techniques developments can be highly qualified software system the user need to be used efficient, reliable and maintainance.

Quality & productivity Factors :-

Quality and productivity are a cirriculum factors in SE. as they directly impact the success of the software project and the satisfaction of end users. Here some of the key Quality factors and productivity factors, They are.

Quality factors :-

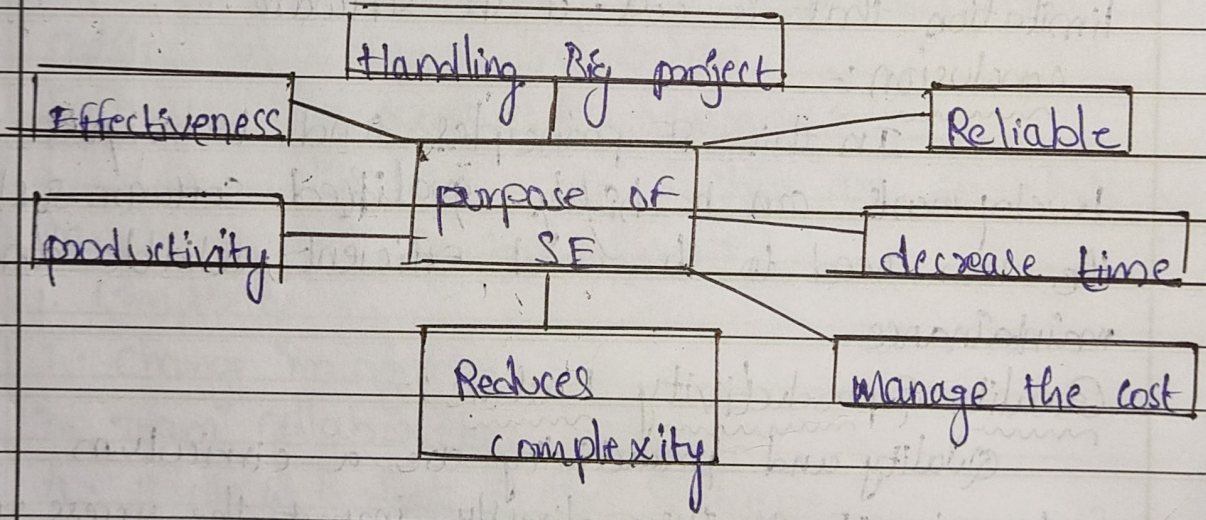
1. Reliability
2. maintainability
3. Useability
4. performance of the software

5. Saurity of the software
6. Flexibility
7. Scalability
8. Documentation.

Productivity factors:-

1. Development
2. Effectiveness
3. Efficient
4. Team collaboration in Software.
5. Tools & Automations
6. Requirement management
7. Continuous integration & Delivery.
8. code quality
9. learning & improvement.

By using this quality & product factors SE teams can be divided and Deliver high quality software products efficiently and effectively maintaining the end users.



Managerial issues:-

In SE managerial issues can be happened after the completion of developing of the software. Improving the principle tools, techniques and process development are shown in the management principles of the SE

principles can be acceptable in an organisation.

→ 1. method of planning, organising, Execution, monitoring, controling and cost of the project.
2. method of cost estimating technique.
3. method for resource allocation policy.
4. Management for Budget control to develop the software.
5. method to setup and maintain the continuous evolution & project scheduling.
6. method for effecting communication between team members and customers.
7. methods to develop efficient control aggrement with customers. So that legal and document can be maintained.

Date: 9/9/24 Page No. 7

Assignment No.: 02

Assignment Topic: planning & development and planning organisation structure

2- Discuss about planning & Development, planning & organisation structure?

A planning & Development process :-

In the software planning and development process is a general outline for planning & Development process in software Engeneering. There are few things to develop a software according to the user requirement the requirement of the software can be specified by user or customer than only the developer of the software can easily analyse what are the requirements of user they are..

1. Requirement Gathering :-
   1. collect the data & document software requirement
   2. Identify the need of the customer.
   3. Define the project scope for developing a project.
2. Designing :-
   1. create a software as a blue print or desiging the
   2. Develop a system architecture which is related to systems.
   3. Design user interface and user Experience.
   4. plan data storage & managing the software.
3. Impdimentation :-
   1. write, clean model structure of coding to develop

a software.
2. use various control system.
3. In implimentation use unit testing & integration testing etc...
4. Testing :-
1. In software plan the testing strategy according to the developer.
2. In testing conduct "system testing & acceptence testing".
3. perform recreation testing & expority testing.
5. Development :-
1. In the software plan the development strategy
2. In this software conduct development testing also
3. In this software the role at the software to protect the development of the software.
6. Maintainance :-
1. The user should plan maintainance activities should follow according to the software.
2. Conduct bug fixing and rectivication of the probdem.
3. perform updates of the software and enhancement of the development of a software.
7. Project management :-
1. plan the project according to the user dead lines which is given by the project time.
2. Manage resources & budject of a software.
3. co-ordinate the team activities & communication of the team members.

Date : ..... Page No. 9

Assignment No. : .....

Assignment Topic : .....

Plan & organisation structure :-

When planning a organisation structure for a software engineering team it's impact to consider a various factors to ensure the "efficient, scalable & goal of the software can be planned and organise of a structure. there are sum of planning and organisation structures.

1. Understanding the key role :-

first understand the key role required in your software engineering team the common role includes

1. product management
2. Engineering management
3. Architecture.
4. Software Engineering.
5. Quality assurance.

2. Team structure models :-

Decide the team structure models that first you organised they are ...

1. Functional team
2. mixed structure
3. future team member.
4. cross functional team members

3. Define clear role & responsibility :-

Ensure that each role has a clear responsibility to help the drawbacks of the system and provides over loop & gaps of the software development come of has to take clear structure

with responsibilities that is
1. product manager
2. Engineering manager
3. Responsibility leader.
4. Software Engineering
5. Architecture.

4. Establishing process & work flow:-
* Define and document the process work flow of your team members. common process includes
1. daily structure plan.
2. Spliting the planning structure both documentation work documentation.
3. focusing on daily report.

5. Communication & collaboration:-
In collaboration & communication choose the proper tools for facilitating for the communication the major components for developing a software there are
1. project management
2. Documentation
3. communication
4. Development.

6. scalability & growth:-
In SE plan for scalability such as
1. Hiring employees properly.
2. Training and development to the employees.
3. Scalable architecture.
4. Distribution. growth.

Date: 23/9/24 Page No. 11

Assignment No.: 03

Assignment Topic: Software cost factors & Estimation techniques

3. Explain about the software cost factors & Estimation techniques?

A. Software cost factor :-

Software cost factors can be very depending on the project, the scope of the project & the environment in which of the development of a software Here some of the common factors for software cost factors.

1. Project size & complexity :-

A project consists of two different sectors large projects & small projects. the large project has more complexity projects, tipically it required more resources to development of the project

2. Requirement :-

the software cost factor changes the requirement during the development of the software process can increase the cost due to reverge & adjustment of the project.

3. technologies :-

The choice of the programming language have different languages like webpage development, frame works and tools can develop the time & cost of the project

4. Skil & experience of development team :-

Highly skilled programming and experience

developers may highly demand their salaries, but they can also more efficiently, potential working process to the development team.

5. Project management:-

The project management has efficient it can help the development process & control the cost estimation.

6. Software development methodologies:-

The development of the software can generate the result in more flexibility & responsibility they can also require more frequent iration to development of a software like extra software to the project.

7. Quality assuring & Testing:-

In the software testing is essential for ensure the software quality and also add to development of cost management system so that quality of the software as well as testing process can be improved.

8. Infrastructure & Environment:-

In the software cost factors are associated with hardware, software icons and to development of different tools and it should be maintain overall project budject or cost estimation of the project.

9. Support & maintainance:-

In the software development cost for the organising support and maintainance as updated should be maintained in the existing budject.

Date : .....

Page No. 13

Assignment No. : .....

Assignment Topic : .....

Estimation Technique :-

Software engineering estimation techniques are essential for planning and managing the software develop -ing of the project. Here some of the common techniques used for various aspects of the software project. The common techniques are.

1. Expert judgment :-

In this techniques expanded innovation or team providency to the judgment of the software based on their experience and knowledge of the similar project. They can be quickly and effectively they can estimate of esspecially for high level planning.

2. Analogous Estimation :-

this method includes using "historical data" from smaller project as a estimating the current project. It assumes that the effort and the duration of new project will be similar to previous projects with comparable of new project.

3. Bottom-up-Estimation :-

This technique involves breaks down the project into smaller, more tasks and more components it is easy to estimate each individuality. The estimat -ion is combine together that is overall project- estimation.

4. Top-down-Estimation:-

Unlike bottom-up-estimation, Top-down-estimation start with an overall estimation for the entire project and then breaks down into smaller components. this method is used for high level planning when detail information is not available.

5. three-point Estimation:-

PERT is a probabilistic estimation technique that consider three estimation for each task, optimistic technique is used in the project. these estimation combine to calculate average estimation to statistical methods.

6. Function point Analysis:-

Function point analysis is a method used to measure the "size and complexity" of software. It assigns the value to each function based on the complexity of the project used to estimation effort, duration and overall cost of the project.

conclusion:-

Each technique has it own strength & weakness and the choice of technique depends on the factors such as project size, complexity, availability of data used a combination of different software development system.

Date: 26/9/24 Page No. 15

Assignment No.: 4

Assignment Topic: SRS & Formal Specification system.

4. Write about SRS & formula specification techniques?

A) A Software Requirement specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform. Here is an outline and explanation of the typical components included in an SRS.

1. Introduction:-

Purpose: Describe the purpose of the SRS and its intended audience.

Scope: Defines the software product to be developed, including its benefits, objectives and goals.

overview:- offers a brief description of the remaining sections of the SRS.

2. Overall Description:-

product perspective: Describe the context and origin of the product, how it interacts with other systems and its user interface.

product functions: Summaries the major functions the software will perform.

Assumptions and Dependencies:- Lits any assumptions made during the development process and any dependencies the project may have.

3. Specific Requirement:-

This section contains all the detailed requirements

necessary for the development of the software. It is often the largest section of the SRS.

- External Interface Requirement :-

Hardware Interface :- Details the interaction with hardware components.

Software Interface : Describe the interactions with other software components.

- Functional Requirements :-

Describe all the functionalities the software must perform. These are usually detailed in terms of use cases or functionali Specifications. Each function includes

→ A description of the function.
→ Inputs and outputs.
→ Details on data handling and processing.

- performance Requirement :-

Specifies the performance criteria for the system, including response time, throughput, availability, and resource utilization.

- Design constraints :-

Describe any design constraints imposed by standards, hardware limitations or other consideration.

- Software system Attributes :-

Reliability : Requirement for the software reliability and availability.

Availability :- when the software should be operational and avdilable for use.

Security :- Requirements for security and data protection.

- Other Requirements :-

Any additional requirements that don't fit into the categories above, such as legal or regulatory requirements.

4- Appendices :

Appendix A : Glossary : Definitions of terms used in SRS.
Appendix B : Analysis model : Diagrams and models used to analyze the system.

Benifits of an SRS :-

- provides a clear understanding of the software's function-alities for all stakeholders.
- serves as a basis for agreement between developers, customers and other stakeholders.
- facilitates communication among project participants.
- provides a foundation for software design and development.
- support project planning and estimation.

Formal specification technique :-

Formal specification techniques all the methods used to describe and analyse the behaviour of the structure of the software system. They provide a understanding development of the software to design problem solving system. such as correctness and complexity of the project by using various techniques.

1. Algebraic specification 2. model based specification

3. process Algebra 4. finite set of missions
5. Executable specification.

Benifits of formal Specification techniques:-

1. verification 4. scalability
2. Documentation. 5. security of the software.
3. consistency

challenges:-

1. complexity :- Critical to develop the software project.
2. tool support :- technologies specified tools.
3. scalability :- may be challenge a very large and complex systems.

Process for requirement specification:-

Requirement specification is a critical case SDLC. Ensure the need and experience team members for developing a software tools. A large and the process can be used with various different development of software development. In some of the common languages were used for the purpose of developing software system the specifications are.

1. natural language (NL)
2. structural natural language.
3. Formal specification language.
4. Simple formal specification language.

Date: 29/9/24

Page No. 19

Assignment No.: 05

Assignment Topic: Design notations & Design Techniques.

5. Explain about the Design notations & techniques.

A. Here's a breakdown of some commonly used design notations in software engineering.

1. UML (unified modeling language):-

- Class Diagram: Represent the static structure of a system, showing classes, attributes, methods, and their relationships.
- Use case Diagram:- Depict interactions between users and a system to achieve specific goals.
- Activity Diagrams:- show the work-flow of a system, representing the flow from one activity to another.

2. flowcharts:-

Useful for representing the flow of control in a system, including decision points, loops and parallel activities.

3. Data flow Diagrams (DFD):-

Depict the flow of data through a system and the process that transform the data.

4. Entity-Relationship Diagrams (ERD):-

Represent the data model for a system, showing entities, attributes, and relationships between entities.

5. Architectural Diagrams:-

Illustrate the high-level structure of a system, including components, their interactions, and dependencies.

6. Dependency structure matrix (DSM) :-
Show dependencies between modules or components in a matrix format, aiding in identifying and managing dependencies.

7 - Components Diagrams :-
Represent the components or modules of a system and their relationships.

8. Deployment Diagrams :-
Show the physical deployment of software components on hardware nodes, illustrating how software and hardware interact in a system.

9. CRC Cards :-
Class - Responsibility - collaboration a technique for.

10. Mockups and wireframes :-
Visual representations of the user interface design, helping to convey the layout and functionality of the system.

These notations can be used individually or in combination, depending on the specific aspects of the system being designed and the preferences of the development team. They serve as valuable tools for documenting, communicating, and analyzing the design of software systems.

Design Techniques :-

Design techniques are aimed at creating robust, scalable, and maintainable software systems. Here are some key design techniques commonly used in software engineering.

1. **Modular Design :-**

Breaking down the software system into smaller, manageble modules or components that can be developed, tested, and maintained independently. This promotes code reusability, scalability, and ease of maintenance.

2. **Object-oriented Design :-**

Organizing software components as objects that encapsulate data and behavior. OOD principles such as inheritance, encapsulation, and polymorphism facilitate modular design and enable easier maintenance and extension of software systems.

3. **Design patterns :**

Reuseable solutions to common design problems encountered during software development. Design patterns provide a structured approach to design and promote best practices for creating flexible and maintainable software.

4- Architectural patterns:-

High-level design patterns that defines the overall structure and organization of software systems. Examples includes the model-view-controller (MVC) pattern for user interfaces and the layered Architecture pattern for organizing system components into layers.

5. Component-Based Design:-

Building software systems by assembling pre-existing software components or modules. component-based design promotes code reuse, accelerates development, and enhances maintainability by isolating changes to individual components.

6. Service-oriented Architecture (SOA):-

Designing software systems as a collection of loosely coupled services that communicate via standardized protocols. SOA promotes reusability, interoperability, and scalability by breaking down complex systems into smaller, interoperable services.

7. Domain-Driven Design (DDD):-

Focusing on the core domain of the problem space and modeling software systems based on domain concepts. DDD emphasizes collaboration between domain experts and software developer to ensure that the software reflects the underlying business domain accurately.

Date: 30/9/24 Page No. 23

Assignment No.: 6

Assignment Topic: real time distributed system & Test plan

6. Explain about the real time and Distributed system Design and test plan?

A. Real time and Distributed system Design:-

Real time and distributed system design encompasses the development of systems that handles tasks concurrently across multiple nodes and adhere to strict timing constraints. Here's a breakdown of key aspect and considerations in designing such systems.

1. Concurrency:-

Real time and distributed systems often involve concurrent execution of tasks across multiple threads, process, or nodes. proper synchronization mechanisms are crucial to ensure data consistency and avoid race conditions.

2. Communication protocols:-

Effective communication between system components is essential in distributed systems. this involves choosing appropriate communication protocols such as TCP/IP, UDP, or message queues, considering factors like latency, reliability, and scalability.

3. Fault tolerance:-

Distributed systems are prone to failure due to network issues, hardware failure, or software bugs. Designing fault-tolerant systems involves implementing redundancy, replication, and error detection

and recovery mechanisms to ensure system reliability and availability.

4. Real-time constraints:-

Real time systems have stringent timing requirements where tasks must be completed within specified deadlines. Designing such systems involves scheduling algorithms, priority assignment, and minimizing latency to ensure timely response to events.

5. Data Replication and consistency:-

Replicating data across distributed nodes improves availability and fault tolerance but introduces challenges in maintaining consistency. Techniques like quorum-based replication and consensus algorithms are used to ensure data consistency in distributed systems.

6. Security:-

Distributed systems are susceptible to security threats such as unauthorized access, data breaches, and denial-of services attacks. Implementing robust security measures including authentication and control is essential to protect sensitive data and ensure system integrity.

7. Monitoring and Debugging:-

Real time monitoring and debugging tools are essential for identifying performance bottlenecks, detecting failures, and analysing system behaviour in distributed environment. This includes logging, tracing and profiling tools to gain insights into system operation and diagnose issue.

Test plans :-

Test plans are crucial components detailing the approach, scope, resources, schedule, and tools required for testing a specific software system or application. Here's a basic structure for a test plan:

1. Introduction :-
   - overview of the document
   - purpose of the test plan
   - Scope of testing
   - References to related documents like requirements specifications, design documents, etc..
2. Test items :-
   - List of software items to be tested.
3. Features to be Tested :-
   - Detailed description of each feature or functionality to be tested.
4. Feature not to be Tested :-
   - mention of any features or functionalities that will not be tested and reasons for exclusion.
   - Test levels
   - Test types.
5. test Deliverables :-
   - list of documentation and artifacts to be delivered as a part of testing.
6. Test schedule :-

- Timeline for testing activities.
- milestones and deliverables.

7. Entry and Exit criteria :-
- conditions to be met before testing can begin
- conditions that indicate the completion of testing

8. Suspension and Resumption Criteria :-
- conditions under which testing may need to be suspended and resumed.

9. Test Risks and contingencies :-
- potential risks to testing activities and their mitigation strategies.

10. Dependencies :-
- Any dependencies that may impact testing activities.

11. Approvals :-
- Sign-off from stakeholders or management.
-

Remember, the level of detail and complexity of a test plan can vary depending on the project's size, complexity, and specific requirements. It's essential to tailor the test plan to suit the needs of your project and organisation.

Date: 4/10/24 Page No. 27

Assignment No.: 7

Assignment Topic: User interface design.

7. Explain about user interface design?

A). User interface design:-

User interface design (UI) is the design of the user interface for the mission and the software such as computers, home applications, mobile devices & other electronic devices, with the focus of software development and user experience. The goal of user interface Design is to make user interaction as simple and effective to solve the problem for the software design.

Real time systems:-

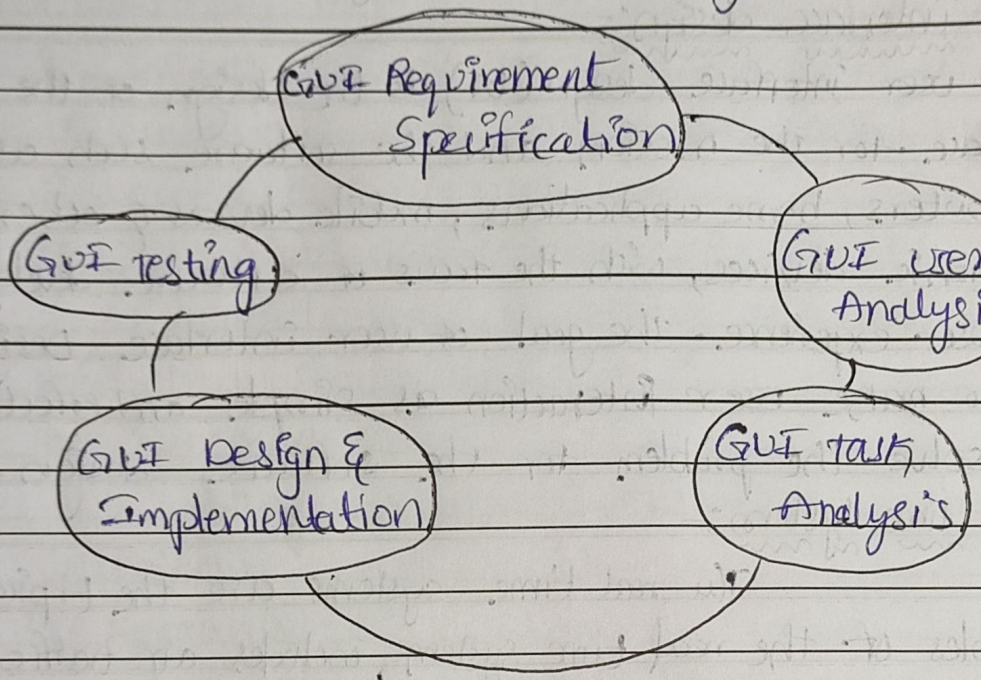
The real time systems are the tipical examples of the real time system includes air traffic control system, networking system, control command system are the real examples in real time distributed system.

The real time system are classified for the specially designed of computer system for a specific purpose commands and control systems, air traffic control system are the examples for the real time system.

User interface system:-

In user interface design there are a no-of activity performed for Designing the user interface. The process of GUI Design and implementation in the SDLC. An any model can be used for GUI.

implimentation among waterfall model, Hydrative model, Spiral model for the design of user interface. A model used for the GUI Design and development of the specific given below the diagram.



There are some of the user interface design

1. GUI Requirement specification:-
This design may like to have a list of all the functional and non functional requirements of GUI. This can be taken from user and their existing software solution.

2. User Analysis:-
The designer studies who is going to use the software GUI. The target audience matters as the design details change according to the knowledge and competency level of the user. If the user is technical savy, advanced and

Complex GUI can be incorporated. For a novice user, more information is included on how-to of software

3. Task Analysis :-

Designers have to analyse what tasks is to be done by the software solution. Here in GUI, it does not matter how it will be done. Task can be represented in hierarchical manner taking one major task and dividing it further into smaller sub-tasks. Tasks provide goals for GUI presentation. Flow of information among subtasks determines the flow of GUI contents in the software.

4. GUI Design & implementation :-

Designers after having information about requirements, tasks and user environment, design the GUI and implements into code and embed the GUI with working or dummy software in the background. It is then self tested by the developers

5. Testing :-

GUI testing can be done in various ways, Organisation can have in house inspection, direct involvement of users and release of beta version are few of them. Testing may include usability, compatibility, user acceptance etc..

Date: 8/10/24 Page No. 31

Assignment No.: 8

Assignment Topic: Human computer interface.

8. Write about human computer interface?

A) Human interface with computers throughout the user interface. This includes software and hardware such as which is displayed on the computer system. with the support of hardware such as external devices like mouse, keyboard and other physical devices which is generated by the result of the system. The human computer interface design is interacting with the different projects between user and the hardware system. usability and experience of the system with the help of HCI Design. There are few types of human computer interface design such as.

1. Usability :-

In this HCI Design is specially designed with the computer system. including efficiency, safety, utility, and other facilities can be used in the software system. The main usability is requirement gathering or resources of the system.

User Experience :-

In the HCI Design focusing on creating the system that are satisfying, entertaining, helpful, motivating to work on more and more experience upon the project the user experience is suddenly work with the various project and

other team members for the development of HCI design.

Interface Design:-

They are a large no. of factors which should be consider in analysis and designing of the system using HCI principles. many of these factors interact with each other, making them analyse even more complex. The main factors are listed in the table as shown below.

1. Organisation factor
2. Environmental factor
3. safety factor
4. continuous factors.
5. comfort factors.
6. user interface factor.
7. task factor.
8. system factor & functionalities.
9. productivity factors.

1. Organisational factors:-

Training, Job design, role of the work, work organisation.

2. Environmental factors:-

Noise of the system, Heating of the system, Lighting of the system.

3. Safety factors:-

the user should have safety and security.

4. continuous processing & capabilities :-
In interface design have motivation, enjoyment, satisfaction, personality and experience.

5. Comfort factors :-
System factors, system settings, Equipment of the system, layout of the system.

6. User interface :-
Input device, output device, icons, commands, Navigation, Graphics, natural languages, user support & multi media.

7. Task factor :-
In this project have easy, complex, task allocation, monitoring and skills of the project.

8. System factor & functionalities :-
Hardware, software, applications.

9. product factors :-
In this project increases the output increases the quality of the software, Decreases the cost of the project, Decreases the error in the project.

Date: 18/10/24

Page No. 35

Assignment No.: 09

Assignment Topic: Software Quality Assurance.

Q. Explain about software Quality Assurance?

A. **Software quality Assurance:-**

Software quality Assurance is a planned to ensure the software products to add the project that is portability, efficiency, reliability and flexibility. It is the collection of activities and functional wedge to monitor and control the software project. so that the specification of the object can be developed in the project.

The software engineering recommands a set of activities for implementing such as.

1. Quality assurance planning.
2. Data gathering.
3. Data Analysis and Reporting.
4. Quality control missions.

"Every software has certain quality goals specified by the customer." these quality goals are to be archived by the development team for a big project.

Software quality Assurance plan:-

The software quality Assurance plan is an ensure the quality of the software levels to develop the project. This plan is used for the development team and quality of the software. The plan provides the frame work and guide lines for the

development of understandable and maintainable code

Quality metrices :-

Software metrices is a standard maintai -nance of the project for the development of measurement It can be classified into three categories.

1. product metrices.
2. process metrices
3. project metrices.

Software testing :-

Software testing is a process of identifying the correctness and quality of the software program. The purpose of software testing satisfy the specific requirement of the system. testing is executing the system or application in order to find the bugs, defects or errors.

Software testing can be the main reason to develop the software and give the advantage to the software developer and make sure that the developer should identify the mistake on the project and the have to regenerate testing code and give the result. there are many ways of software testing.

1. Manual testing :-

test case executed manually.

2. Automation testing :-

Testing performed with help of automation tools.