V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses

📞 **+91 93260 50669**

🌐 **v2vedtech.com**

▶️ **V2V EdTech LLP**

📷 **v2vedtech**

_____

WINTER – 2022 EXAMINATION

Subject Name: Software Engineering	Model Answer	Subject Code: 22413

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a)	**State the characteristics of Software Engineering.**	**2 M**
	Ans	<u>**Characteristics of software engineering are :**</u> 1. Software is developed or engineered; it is not manufactured in the classical sense. 2. Software doesn't "wear out." 3. Although the industry is moving toward component-based construction, most software continues to be custom built.	**Any 2 characteristics=2 M**
	b)	**Define : i) Software ii) Software Engineering**	**2 M**
	Ans	**Software:** Software is: 1. Instructions (computer programs) that when executed provide desired features, function, and performance; 2. Data structures that enable the programs to adequately manipulate information, and 3. Descriptive information (documents) in both hard copy and virtual forms that describes the operation and use of the programs. **Software Engineering:** Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.	**Software definition: 1 M; Software Engineering definition: 1 M or any other relevant definition shall be given marks**

c)	**State the characteristics of SRS.**		**2 M**
Ans	**Characteristics of SRS are :** • Correct • Complete • Unambiguous • Verifiable • Consistent • Ranked for importance and/or stability • Modifiable • Traceable		**any 4 characteristics of SRS : 2 M**
d)	**List the project cost Estimation Approaches.**		**2 M**
Ans	**Project cost Estimation Approaches are :** 1. Heuristic Estimation Approach 2. Analytical Estimation Approach 3. Empirical Estimation Approach		**Any two project cost Estimation Approaches : 2 M ; 1 M each**
e)	**Define risk and list any two types of risk.**		**2 M**
Ans	**Risk:** A risk is "an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives." **OR** Risk is the uncertainty which is associated with a future event which may or may not occur and a corresponding potential for loss. Types of risks are : 1. Generic risk 2. Product specific risk **OR** 1.Schedule / Time-Related / Delivery Related Planning Risks 2. Budget / Financial Risks 3. Operational / Procedural Risks 4. Technical / Functional / Performance Risks 5. Other Unavoidable Risks		**Risk : definition: 1 M;** **any two types of risks : 1 M**
f)	**Define Software Quality Control and Quality Assurance.**		**2 M**
Ans	**Software Quality Control**: It is a procedure that focuses on fulfilling the quality requested. **Quality Assurance:** It is a procedure that focuses on providing assurance that quality requested will be achieved. Quality assurance consists of the auditing and reporting functions of management.		**Definition of Software Quality Control : 1 M and Quality Assurance : 1 M** **(any other relevant definitions should**

_____

			be given marks)
	g)	List the phases of Software Quality Assurance.	**2 M**
	Ans	**Phases of Software Quality Assurance are :** • SQA Planning. • Activities. • Review and Audit.	**List of phases of Software Quality Assurance : 2 M**
2.		**Attempt any THREE of the following:**	**12 M**
	a)	**State and describe any four types of Software.**	**4 M**
	Ans	**1. System software:** System software is a collection of programs written to service other programs. Some system software (e.g., compilers, editors, and file management utilities) process complex, but determinate, information structures. Other systems applications (e.g., operating system components, drivers, telecommunications processors) process largely indeterminate data. In either case, the system software area is characterized by heavy interaction with computer hardware; heavy usage by multiple users; concurrent operation that requires scheduling, resource sharing, and sophisticated process management; complex data structures; and multiple external interfaces. **2. Real-time software:** Software that monitors/analyzes/controls real-world events as they occur is called real time. Elements of real-time software include a data gathering component that collects and formats information from an external environment, an analysis component that transforms information as required by the application, a control/output component that responds to the external environment, and a monitoring component that coordinates all other components so that real-time response (typically ranging from 1 millisecond to 1 second) can be maintained. **3. Business software:** Business information processing is the largest single software application area. Discrete "systems" (e.g., payroll, accounts receivable/payable, inventory) have evolved into management information system (MIS) software that accesses one or more large databases containing business information. Applications in this area restructure existing data in a way that facilitates business operations or management decision making. In addition to conventional data processing application, Business software applications also encompass interactive computing (e.g., point-of-sale	**Stating and describing 4 types of software : 4 M; 1 M each**

transaction processing).

4. Engineering and scientific software: Engineering and scientific software have been characterized by "number crunching" algorithms. Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing. However, modern applications within the engineering/scientific area are moving away from conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take on real-time and even system software characteristics.

5. Embedded software: Intelligent products have become commonplace in nearly every consumer and industrial market. Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets.
Embedded software can perform very limited and esoteric functions (e.g., keypad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems).
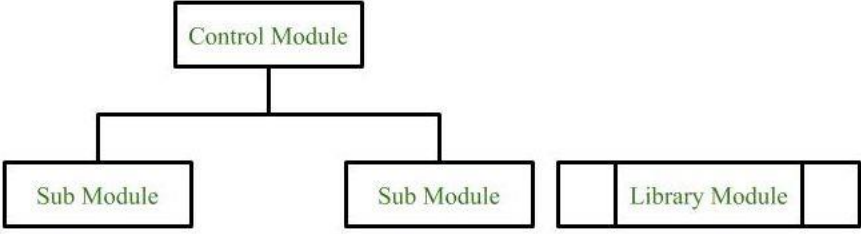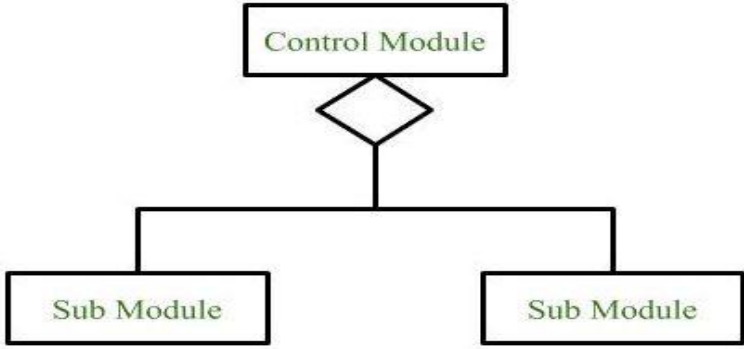
6. Product line or Personal computer software: The personal computer software market has burgeoned over the past two decades. Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications, external network, and database access are only a few of hundreds of applications.
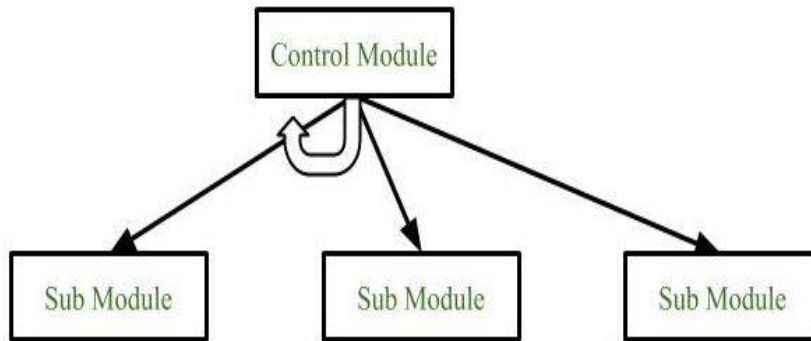
7. Web-based software: The Web pages retrieved by a browser are software that incorporates executable instructions (e.g., CGI, HTML, Perl, or Java), and data (e.g., hypertext and a variety of visual and audio formats). In essence, the network becomes a massive computer providing an almost unlimited software resource that can be accessed by anyone with a modem.

8. Artificial intelligence software: Artificial intelligence (AI) software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Expert systems, also called knowledge-based systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing are representative of applications within this category.

|			
	b)	**Explain structured flowchart with suitable example.**	**4 M**
	An	**Structured flowchart** represents hierarchical structure of modules. It breaks down the entire system into lowest functional modules; describe functions	**Structured flowchart with**

s	and sub-functions of each module of a system to a greater detail. Structured flowchart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown). Inputs are given to the black boxes and appropriate outputs are generated. Modules at top level called modules at low level. Components are read from top to bottom and left to right. When a module calls another, it views the called module as black box, passing required parameters and receiving results.	**example : 4 M** **Or** **any other relevant structured flowchart with example give 4M**

Symbols used in construction of structured chart

1. **Module**

   It represents the process or task of the system. It is of three types.

   - **Control Module**
     A control module branches to more than one sub module.
   - **Sub Module**
     Sub Module is a module which is the part (Child) of another module.
   - **Library Module**
     Library Module are reusable and invokable from any module.

2. **Conditional Call**

   It represents that control module can select any of the sub module on the basis of some condition.

3. **Loop (Repetitive call of module)**

   It represents the repetitive execution of module by the sub module.
   A curved arrow represents loop in the module.

All the sub modules cover by the loop repeat execution of module.
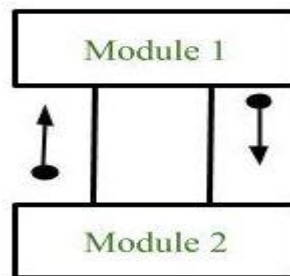
4. **Data Flow**
   It represents the flow of data between the modules. It is represented by directed arrow with empty circle at the end.
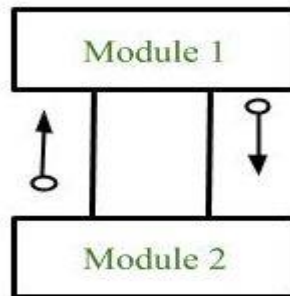
5. **Control Flow**
   It represents the flow of control between the modules. It is represented by directed arrow with filled circle at the end.
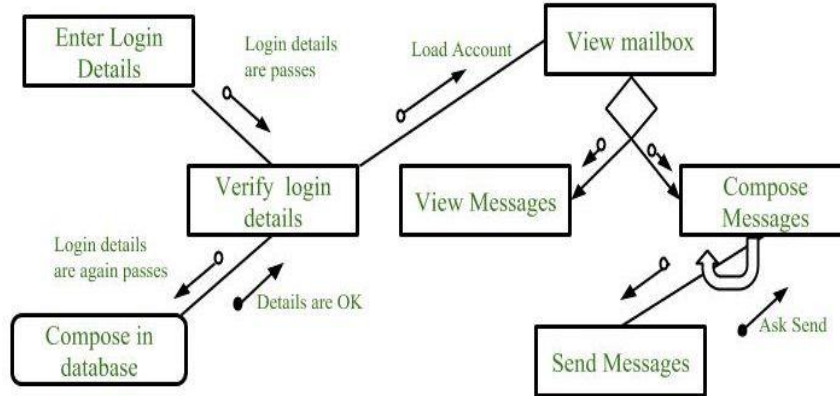

6. **Physical Storage**
   Physical Storage is that where all the information are to be stored.

		Example : Structure chart for an Email server	
	c)	**Describe 4P's of management spectrum.**	**4 M**
	An s	The management spectrum focuses on the four P_s; people, product, process and project. 1. **The People:** • People of a project includes from manager to developer, from customer to end user .But mainly people of a project highlight the developers. • It is so important to have highly skilled and motivated developers that the Software Engineering Institute has developed a People Management Capability Maturity Model (PM-CMM), • Organizations that achieve high levels of maturity in the people management area have a higher likelihood of implementing effective software engineering practices. 2. **The Product:** • The product is the ultimate goal of the project. • This is any types of software product that has to be developed. • To develop a software product successfully, all the product objectives and scopes should be established, alternative solutions should be considered, and technical and management constraints should be identified beforehand. • Lack of these information, it is impossible to define reasonable and accurate estimation of the cost, an effective assessment of risks, a realistic breakdown of project tasks or a manageable project schedule that provides a meaningful indication of progress. 3. **The Process:** • A software process provides the framework from which a comprehensive plan for software development can be established.	**4 P's of management spectrum : 4 M; 1 M each**

		• A number of different tasks sets— tasks, milestones, work products, and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. • Finally, umbrella activities overlay the software process model. • Umbrella activities are independent of any one framework activity and occur throughout the process. 4. **The Project:** • The project is the complete software project that includes requirement analysis, development, delivery, maintenance and updates. • The project manager of a project or sub-project is responsible for managing the people, product and process. • The responsibilities or activities of software project manager would be a long list but that has to be followed to avoid project failure. • A software project could be extremely complex and as per the industry data the failure rate is high. • Its merely due to the development but mostly due to the steps before development and sometimes due to the lack of maintenance.	
d)		Describe critical path method with suitable example.	**4 M**
Ans		**CPM:-** (a) A critical path in project management is certain tasks that need to be performed in a clear order and for a certain period. (b) If part of one task can be slowed down or postponed for a term without leaving work on others, then such a task is not critical. (c) While tasks with a critical value cannot be delayed during the implementation of the project and are limited in time. (d) Critical Path Method (CPM) is an algorithm for planning, managing and analyzing the timing of a project. (e) The step-by-step CPM system helps to identify critical and non-critical tasks from projects' start to completion and prevents temporary risks. (f) Critical tasks have a zero run-time reserve. If the duration of these tasks changes, the terms of the entire project will be "shifted." That is why critical tasks in project management require special control and timely detection of risks. (g) The method was developed by one of the American companies in 1957. Its employees planned to close, repair and restart chemical plants. (h) The tasks in this project were numerous and complex; that's why they required such a method. (i) After that, Critical Path Method was quickly spread to agricultural and construction projects where people wanted to learn how to avoid routine tasks.	**Critical path method with suitable example : 4 M**

(j) Today, this method of identifying critical tasks is widely used in many industries, including software development.
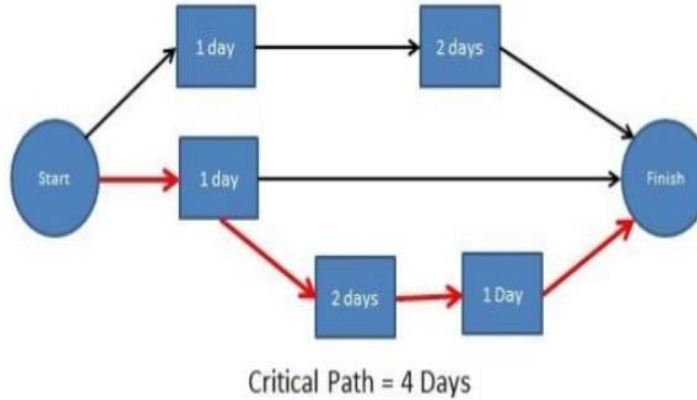
Example :

Critical Path = 4 Days

Figure 3: Critical Path Method.

3.		Attempt any **THREE** of the following:	12 M
	a)	Distinguish between waterfall model and spiral model.	4 M
	Ans	**Difference Waterfall and Spiral Model:**	Any 4: 1 M each

Parameters	Waterfall Model	Spiral Model
Working Nature	Sequential method	Evolutionary method
Involvement of Customer	Minimum, at end of project completion	Maximum, earlier in project development
Identification & rectification of errors	After completion of all stages	Earlier while developing the project
Simplicity	Easy Model	Complex Model
Flow of the Phases	One after another, difficult to go back	In iteration, easy to go back
Project size	Small	Large
Flexibility to change contents	Difficult	Easy
Cost	Less	More
Framework type	Linear	Iterative

	b)	Describe any four Software analysis-modeling principles.	4 M
	Ans	**Four Software analysis modeling principles:** 1. **Principle 1:** The information domain of a problem must be represented and understood.	Any 4: 1 M each

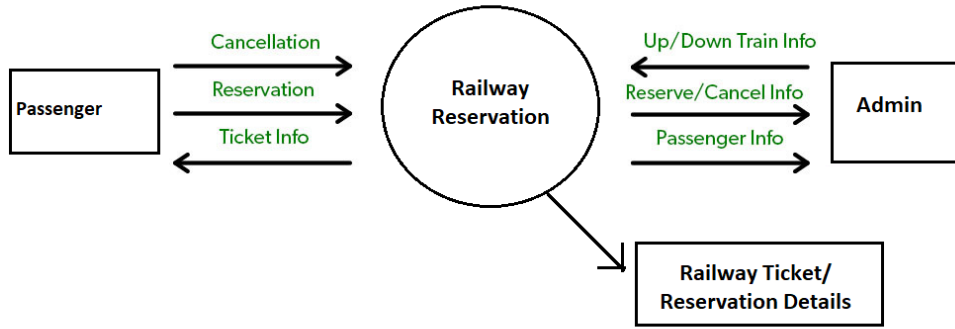|			
		The information domain encompasses the data that flow into the system, the data that flow out of the system, and the data stores that collect and organize persistent data objects. **2. Principle 2:** The functions that the software performs must be defined. Software functions provide direct benefit to end users and also provide internal support for those features that are user visible. Some functions transform data that flow into the system. In other cases, functions affect some level of control over internal software processing or external system elements. Functions can be described at many different levels of abstraction, ranging from a general statement of purpose to a detailed description of the processing elements that must be invoked. **3. Principle 3:** The behavior of the software must be represented. The behavior of computer software is driven by its interaction with the external environment. Input provided by end users, control data provided by an external system, or monitoring data collected over a network all cause the software to behave in a specific way. **4. Principle 4:** The models that depict information function and behavior must be partitioned in a manner that uncovers detail in a layered (or hierarchical) fashion. Requirement's modeling is the first step in software engineering problem solving. It allows you to better understand the problem and establishes a basis for the solution. Complex problems are difficult to solve in their entirety. For this reason, you should use a divide and-conquer strategy. A large, complex problem is divided into sub problems until each sub problem is relatively easy to understand. This concept is called partitioning or separation of concerns, and it is a key strategy in requirements modeling. **5. Principle 5:** The analysis task should move from essential information toward implementation detail. Requirements modeling begin by describing the problem from the end-user 's perspective. The essence of the problem is described without any consideration of how a solution will be implemented.	
	c)	**Draw DFD for Railway Reservation Management System for level 0 and level 1.**	**4 M**
	An s	**Level 0 DFD: railway Reservation System:**	**2 M for Level 0 and 2M for Level 1** **OR** **any other relevant Level o and 1 shall**

be given marks

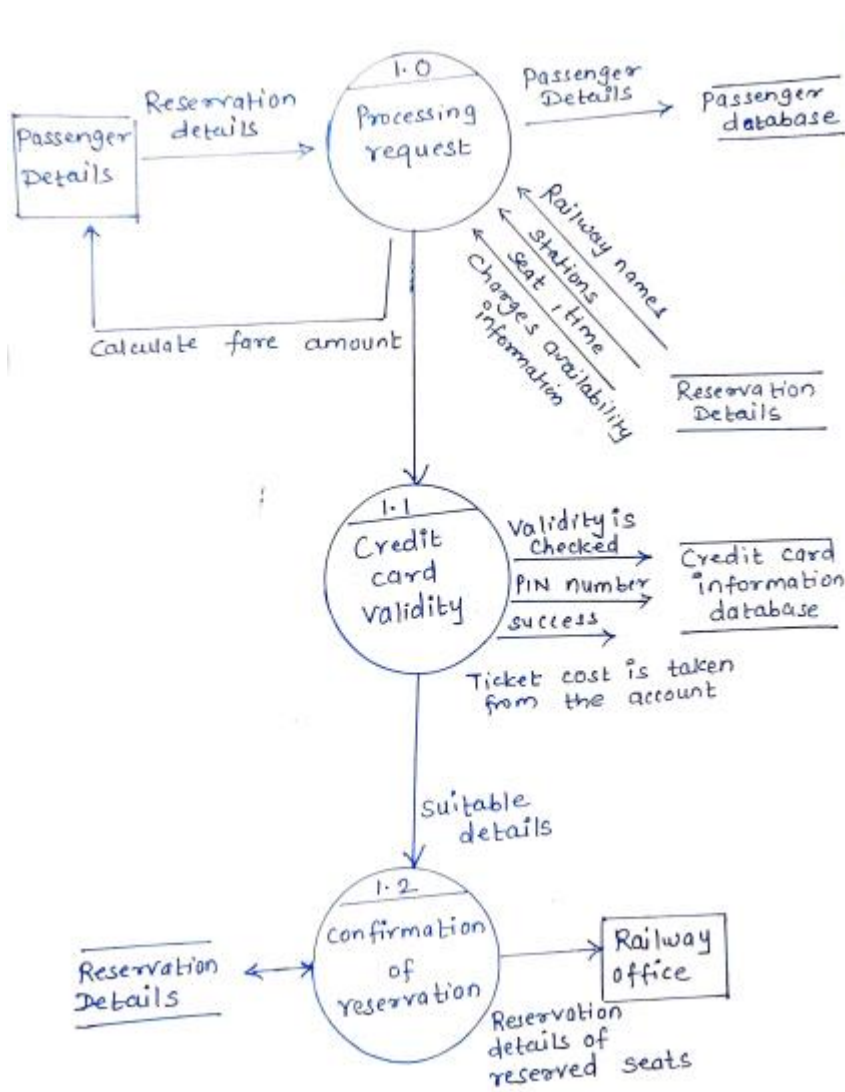**Fig: Level 0 DFD: Railway Reservation System**

Fig: Level 1 DFD: Railway Reservation System

d)	Explain line of code metrics for size estimation.	4M

Ans	Line of code metrics for size estimation: LOC count the total number of lines of source code in a project.
	
	The units of LOC are:
	    KLOC- Thousand lines of code
	    NLOC- Non-comment lines of code
	    KDSI- Thousands of delivered source instruction
	
	The size is estimated by comparing it with the existing systems of the same kind. The experts use it to predict the required size of various components of software and then add them to get the total size.
	
	Parameters to count LOC:
	    1. count only executable lines.
	    2. count executable lines plus data definitions.
	    3. count executable lines, data definitions and comments.
	    4. count physical lines on input screen.
	
	Consider the following example for counting LOC:
	    KCSI: thousands changed source instructions.
	    KSSI: thousands shipped source instructions.
	
	First Release of Product Y
	    KCSI = KSSI = 50 KLOC
	    Defects/KCSI = 2.0
	    Total number of defects = $2.0 \times 50 = 100$
	
	Second Release,
	    KCSI = 20 KSSI = 50+ 20 (new and changed lines of code) -4 (assuming 20% are changed lines of code) = 66
	
	Defect/KCSI = 1.8 (assuming 10% improvement over the first release). Total number of additional defects = 1.8 x 20 = 36.
	
	Third Release,
	    KCSI=30
	    KSSI 66+30 (new and changed lines of code) -6 (assuming 20% of changed lines of code) = 90.
	Targeted number of additional defects (no more than previous release) = 36. Defect rate target for the new and changed lines of code: 36/30= 1.2 defects/KCSI or lower.
		Proper explanation=4M

		Advantages: 1)Universally accepted and is used in many models like COCOMO. 2)Estimation is closer to the developer's perspective. **Disadvantages:** 1)Different programming languages contain a different number of lines. 2)No proper industry standard exists for this technique. 3)It is difficult to estimate the size using this technique in the early stages of the project.	
4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Explain Dynamic Systems Development Method (DSDM).**	**4 M**
	An s	**Dynamic Systems Development Method (DSDM):** **Dynamic Systems Development Method life cycle** 1. **Feasibility Study:** It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be a viable candidate for the DSDM method. 2. **Business Study:** It establishes the use and knowledge necessities that may permit the applying to supply business value; additionally, it is the essential application design and identifies the maintainability necessities for the applying. 3. **Functional Model Iteration:** It produces a collection of progressive prototypes that demonstrate practicality	**1 M for diagram** **3M for explanation**
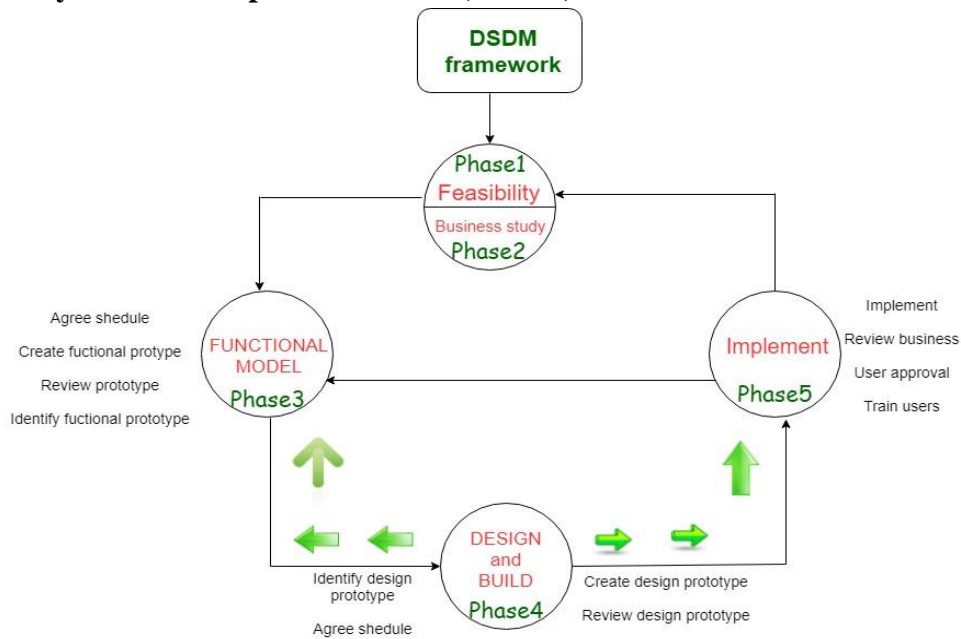
_____

		for the client. (Note: All DSDM prototypes are supposed to evolve into the deliverable application.) The intent throughout this unvarying cycle is to collect further necessities by eliciting feedback from users as they exercise the paradigm. 4. **Design and Build Iteration:** It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may alter it to supply operational business price for finish users. In some cases, useful model iteration and style and build iteration occur at the same time. 5. **Implementation:** It places the newest code increment (an "operationalized" prototype) into the operational surroundings. It ought to be noted that: **(a)** the increment might not 100% complete or, **(b)** Changes are also requested because the increment is placed into place. In either case, DSDM development work continues by returning to the useful model iteration activity.	
b)		**State software engineering practices and its importance.**	**4 M**
Ans		**Software Engineering practices and its importance:** <u>**Software Engineering Practices:**</u> 1. Understand the problem (communication and analysis). 2. Plan a solution (modeling and software design). 3. Carry out the plan (code generation). 4. Examine the result for accuracy (testing and quality assurance). **Understand the problem:** • Who has a stake in the solution to the problem? That is, who are the stakeholders? • What are the unknowns? What data, functions, features, and behavior are required to properly solve the problem? • Can the problem be compartmentalized? Is it possible to represent smaller problems that may be easier to understand? Can the problem be represented graphically? Can an analysis model be created? **Plan the solution:** • Have you seen similar problems before? Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, features, and behavior that are required? • Has a similar problem been solved? If so, are solutions readily apparent for the sub-problems? • Can you represent a solution in a manner that leads to effective implementation? Can a design model be created? **Carry out the plan:** • Does the solution confirm to the plan? IS source code traceable to the design	**Software Engineering practices=2M** **(any 2 Points)** **and** **Software Engineering importance=2M** **(any 2 Points)**

model?
• Is each component part of the solution probably correct? Have the design and code been received, or better, has correctness proof been applied to the algorithm?

Examine the result:
• Is it possible to test each component part of the solution? Has a reasonable testing strategy been implemented?
• Does the solution produce results that confirm to the data? Functions, features and behavior that are required? Has the software been validated against all stakeholder requirements?

<u>Importance of Software Engineering:</u>

The importance of software engineering lies in the fact that a specific piece of Software is required in almost every industry, every business, and purpose. As time goes on, it becomes more important for the following reasons.

1. Reduces Complexity

Dealing with big Software is very complicated and challenging. Thus to reduce the complications of projects, software engineering has great solutions. It simplifies complex problems and solves those issues one by one.

2. Handling Big Projects

Big projects need lots of patience, planning, and management, which you never get from any company. The company will invest its resources; therefore, it should be completed within the deadline. It is only possible if the company uses software engineering to deal with big projects without problems.

3. To Minimize Software Costs

Software engineers are paid highly as Software needs a lot of hard work and workforce development. These are developed with the help of a large number of codes. But programmers in software engineering project all things and reduce the things which are not needed. As a result of the production of Software, costs become less and more affordable for Software that does not use this method.

4. To Decrease Time

If things are not made according to the procedures, it becomes a huge loss of time. Accordingly, complex Software must run much code to get definitive running code. So it takes lots of time if not handled properly. And if you follow the prescribed software engineering methods, it will save your precious time by decreasing it.

5. Effectiveness

Making standards decides the effectiveness of things. Therefore a company

_____

		always targets the software standard to make it more effective. And Software becomes more effective only with the help of software engineering. **6. Reliable Software** The Software will be reliable if software engineering, testing, and maintenance are given. As a software developer, you must ensure that the Software is secure and will work for the period or subscription you have agreed upon.	
	c)	**State and explain the component of Risk Management.**	**4 M**
	An s	**Component of Risk Management:** Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project.	**Any 4 components=** **4M**

Fig: Components of Risk Management

Components:
1. Risk Identification
Risk identification is the process of documenting potential risks and then categorizing the actual risks the business faces.
When identifying risk, it's also important to not just think about the risks that the business currently faces, but those that might emerge in the future, as well.

2. Risk Analysis
Once risks have been identified, the next step is to analyze their likelihood and potential impact.
How exposed is the business to a particular risk? What is the potential cost of a risk becoming a reality?
An organization might divide risks into "serious, moderate, or minor" or "high, medium, or low" depending on their potential for disruption.

_____

		3. Response Planning Response planning answers the question: What are we going to do about it? For example, if during identification and analysis, you realized that the business is at risk of phishing attacks because its employees are unaware of email security best practices, your response plan might include security awareness training. **4. Risk Mitigation** Risk mitigation is the implementation of your response plan. It is the action your business and its employees take to reduce exposure. **5. Risk Monitoring** Risks are not static; they change over time. Risk monitoring is the process of "keeping an eye" on the situation through regular risk assessments.	
	d)	**Describe following project cost estimation approaches** **i) Heuristic** **ii) Empirical**	**4 M**
	Ans	**Project cost estimation approaches** **1. Empirical Estimation Technique:** Empirical estimation is a technique or model in which empirically derived formulas are used for predicting the data that are a required and essential part of the software project planning step. These techniques are usually based on the data that is collected previously from a project and also based on some guesses, prior experience with the development of similar types of projects, and assumptions. It uses the size of the software to estimate the effort. In this technique, an educated guess of project parameters is made. Hence, these models are based on common sense. However, as there are many activities involved in empirical estimation techniques, this technique is formalized. **2. Heuristic Technique:** Heuristic means "to discover". The heuristic technique is a technique or model that is used for solving problems, learning, or discovery in the practical methods which are used for achieving immediate goals. These techniques are flexible and simple for taking quick decisions through shortcuts and good enough calculations, most probably when working with complex data. But the decisions that are made using this technique are necessary to be optimal. In this technique, the relationship among different project parameters is expressed using mathematical equations. The popular heuristic technique is given by Constructive Cost Model (COCOMO). This technique is also used to increase or speed up the analysis and investment decisions.	**2M for Heuristic and 2 M for Empirical**

	e)	Prepare macro time line chart for 15 days of college management system (5 days a week) consider phases of SDLC.	4 M
	Ans	**Time Chart:** **Fig:Time line chart for 15 days of college management system**	Correct Time line chart=4M
5.		Attempt any **TWO** of the following:	12 M
	a)	Sketch use-case diagram for Library management with minimum four use cases and two actors.	6 M
	Ans	Use-case diagram for Library management.	Correct Diagram for any four use cases and Actor =6M
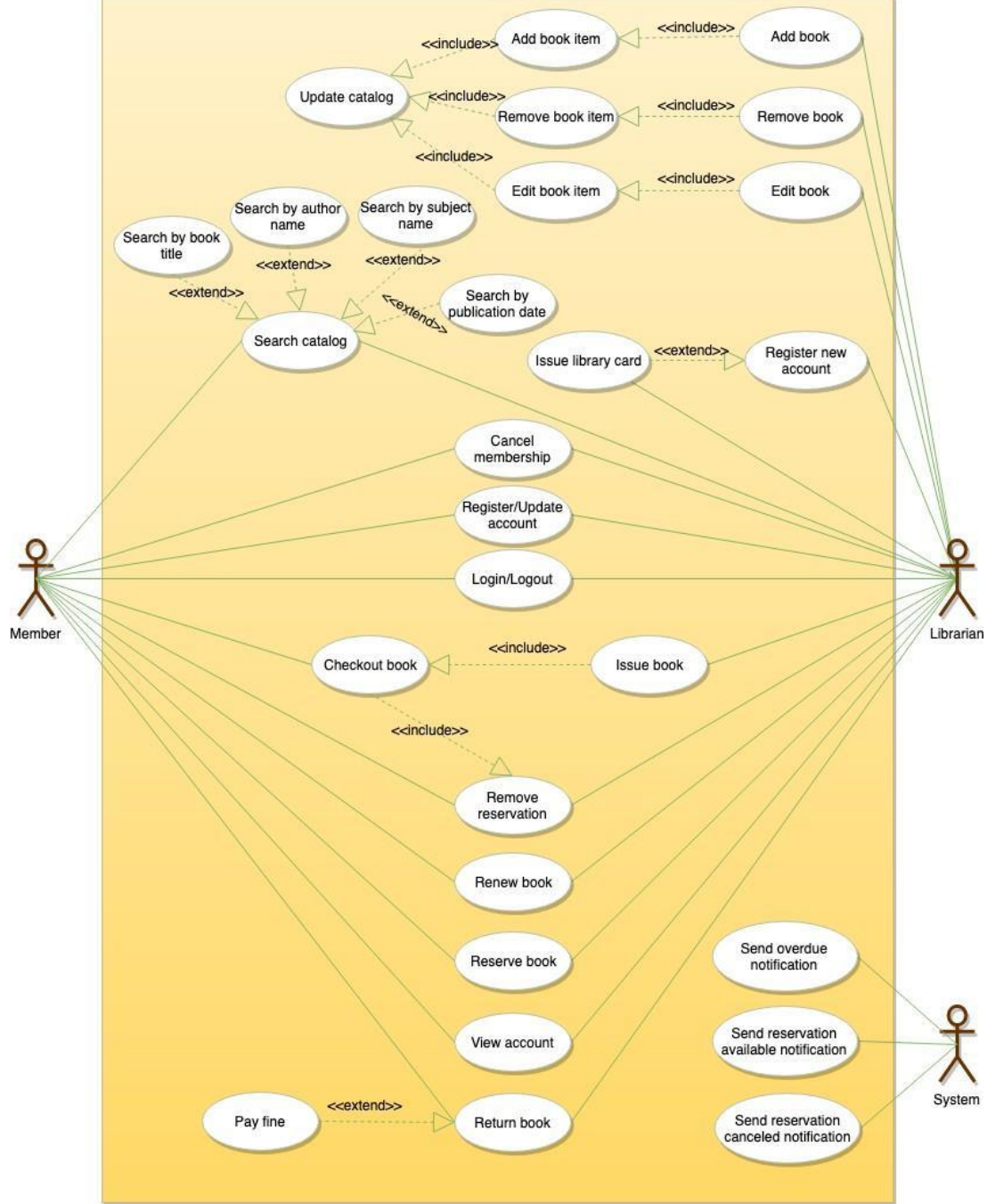
| b) | Differentiate between white box testing and black box testing. | 6 M |

	Sr.no	White box testing	Black Box Testing	6 M
Ans				1M = 1 Point
	1	The tester needs to have the knowledge of internal code or program.	This technique is used to test the software without the knowledge of internal code or program.	
	2	It aims at testing the structure of the item being tested.	It aims at testing the functionality of the software.	
	3	It is also called structural testing, clear box testing, code-based testing, or glass box testing.	It also knowns as data- driven, box testing, data and functional testing.	
	4	Testing is best suited for a lower level of testing like Unit Testing, Integration testing.	This type of testing is ideal for higher levels of testing like System Testing, Acceptance testing.	
	5	Statement Coverage, Branch coverage, and Path coverage are White Box testing technique.	Equivalence partitioning, Boundary value analysis are Black Box testing technique	
	6	Can be based on detailed design documents.	Can be based on Requirement specification document.	
	7	Example: By input to check and verify loops	Example: Search something on google by using keywords	

c)	**Describe a Cocomo and Cocomo-lI models.**		**6 M**

Ans	**COCOMO Model :**
	
	 • COCOMO is one of the most widely used software estimation models in the world.
	 • This model is developed in 1981 by Barry Boehm to give estimation of number of man-months it will take to develop a software product.
	 • COCOMO predicts the efforts and schedule of software product based on size of software.
	 • COCOMO has three different models that reflect complexity
	
	   1. Basic Model
	   2. Intermediate Model
	   3. Complete Model
			3 M for Explanation of COCOMO Model
			
			 and
			
			 3 M for Explanation of COCOMO Model-II

1) **Basic COCOMO:** The basic COCOMO is employed for rough calculations, limiting software estimation precision. This is because the model only considers lines of source code and constant values derived from software project types rather than other elements that significantly impact the software development process.

2) **Intermediate COCOMO:** The Intermediate COCOMO model expands the Basic COCOMO model that takes into account a collection of cost drivers to improve the cost estimating model's accuracy.

3) **Complete/Detailed COCOMO:** The model contains all qualities of both Basic COCOMO and Intermediate COCOMO techniques for each software engineering process. The model considers each project's development phase (analysis, design, and so on).

Estimation of Effort: Calculations –
  Basic Model gives an approximate estimation of the project parameter.
  The Basic COCOMO Estimation model given by following Expression ,

$$E = a(KLOC)^b$$

 The Cocomo model divides software projects into 3 types-

1. Organic Project
It belongs to small & simple software projects which are handled by a small team with good domain knowledge and few rigid requirements.
Example: Small data processing or Inventory management system.

2. Semidetached Project
It is an intermediate (in terms of size and complexity) project, where the team having mixed experience (both experience & inexperience resources) to deals with rigid/nonrigid requirements.
Example: Database design or OS development.

3. Embedded Project
This project having a high level of complexity with a large team size by considering all sets of parameters (software, hardware and operational).
Example: Banking software or Traffic light control software.

COCOMO II Model :

		COCOMO-II is the revised version of the original Cocomo (Constructive Cost Model) and is developed at University of Southern California. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.	
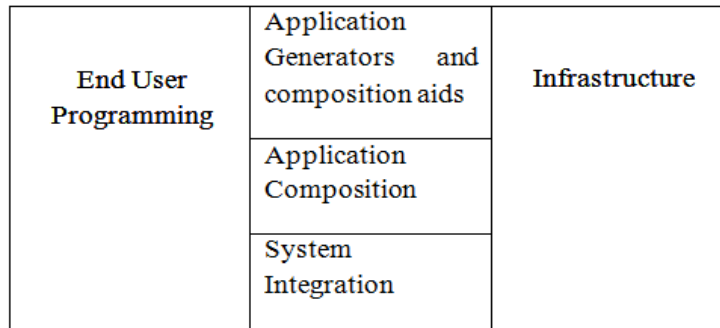
- The Application Composition Model

    This model involves prototyping efforts to resolve potential high- risk issues such as user interfaces, software/system interaction, performance, or technology maturity. The costs of this type of effort are best estimated by the Applications Composition model. It is suitable for projects built with modern GUI-builder tools. It is based on new Object Points.

- The Early Design Model

    The Early Design model involves exploration of alternative software/system architectures and concepts of operation. It uses a small set of new Cost Drivers, and new estimating equations. Based on Unadjusted Function Points or KSLOC.
- The Post-Architecture Model

    The Post-Architecture model involves the actual development and maintenance of a software product Estimates. In COCOMO II effort is expressed as Person Months (PM). The inputs are the Size of software development, a constant, A, and a scale factor, B. The size is in units of thousands of source lines of code (KSLOC). The constant, A, is used to capture the multiplicative effects on effort with projects of increasing size.

6.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Recognize requirement for following modules of hospital management software** **i) Customer module** **ii) Administrator module** **iii) Account module**	**6 M**
	An	Requirement for following modules of hospital management software:	**6M=2M for each**

s	**i) Customer module** a) Services provided in hospital b) Facility to register patients and view their report and history. c) Avaibility of beds and wards etc. d) Showing Dr qualification, avaibility for OPD. **ii) Administrator module** a)Administrator can view as well as alter ant information of the Hospital Management System b) Authority to all purchase and employee Management. c) Updating Availability beds and wards. d) Add patients details and assigning ID. **iii) Account module** a) Details about patients Health Insurance. b) Bill Generation. c) Develop maintains and analyses budgets, preparing periodic reports that compare budgeted costs to actual costs. d) Oversee the Hospitals Billing Department.		**Module**
b)	**Draw neat labelled diagram of translation of requirement model in to design model and explain it with details.**		**6 M**
Ans	**Translation of Requirement model into design model :** Software requirements, manifested by the data, functional, and behavioral models, feed the design task. Using one of a number of design methods, the design task produces a data design, an architectural design, an interface design, and a component design. Each of the elements of the analysis model provides information that is necessary to create the four design models required for a complete specification of design.		**2M : Diagram** **4M : Explanation**

		Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer's requirements and at the same time assessed for quality against a set of predefined criteria for —good‖ design. In the software engineering context, design focuses on four major areas of concern: data, architecture, interfaces, and components Design begins with the requirements model. The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software. The data objects and relationships defined in the entity relationship diagram and the detailed data content depicted in the data dictionary provide the basis for the data design activity. Part of data design may occur in conjunction with the design of software architecture. More detailed data design occurs as each software component is designed. The architectural design defines the relationship between major structural elements of the software, the design pattern that can be used to achieve the requirements that have been defined for the system, and the constraints that affect the way in which architectural design patterns can be applied. The architectural design representation the framework of a computer based system can be derived from the system specification, the analysis model, and the interaction of subsystems defined within the analysis model. The interface design describes how the software communicates within itself, with systems that interoperate with it, and with humans who use it. An interface implies a flow of information (e.g., data and/or control) and a specific type of behavior. Therefore, data and control flow diagrams provide much of the information required for interface design. The component-level design transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the PSPEC, CSPEC, and STD serve as the basis for component design.	
	c)	**Explain CMMI Techniques with its level.**	**6 M**
	Ans	**CMMI Techniques :** The Capability Maturity Model Integration (CMMI), a comprehensive process meta-model that is predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and maturity. The CMMI represents a process meta-model in two different ways: (1) Continuous model and (2) Staged model. The continuous CMMI meta-model describes a process in two dimensions. Each process area (e.g. project planning or requirements management) is formally assessed against specific goals and practices and is rated according to the following capability levels.	**1M : diagram ,** **5M : Any 5 Point**

Level 1: Initial. The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

Level 2: Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Level 3: Defined. The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process for developing and supporting software. This level includes all characteristics defined for level 2

Level 4: Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level 3

Level 5: Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all characteristics defined for level 4.