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

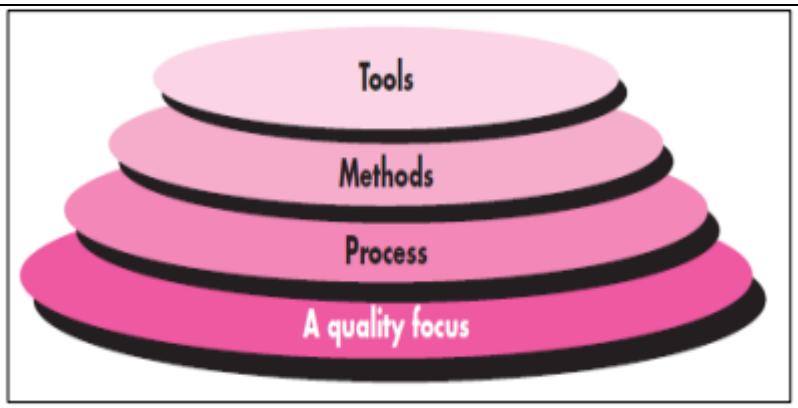
v2vedtech

SUMMER – 2023 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

Subject Name: Software Engineering **Subject Code:** 22413

Important Instructions to examiners:

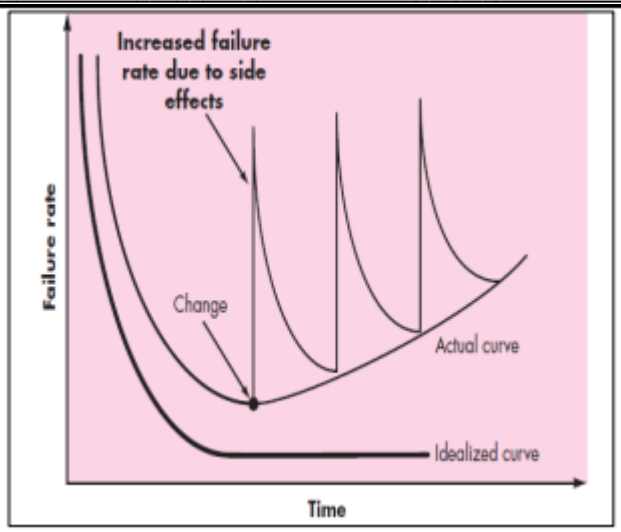
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling 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)	**Draw layered approach of software engineering.**	**2 M**
	Ans		Correct Diagram -2 M
	b)	**Define software engineering.**	**2 M**
	Ans	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.	Correct Definition-2 M

<table>
<tr><td></td><td>c)</td><td>List any four characteristics of good SRS.</td><td>2 M</td></tr>
<tr><td></td><td>Ans</td><td>Characteristics of SRS are:
• Correct
• Complete
• Unambiguous
• Verifiable
• Consistent
• Ranked for importance and/or stability
• Modifiable
• Traceable</td><td>Any 4 correct characteristics of SRS: 2 M</td></tr>
<tr><td></td><td>d)</td><td>Enlist four types of risks.</td><td>2 M</td></tr>
<tr><td></td><td>Ans</td><td>Types of risks are:
1. Generic risk
2. Product specific risk
3. Schedule / Time-Related / Delivery Related Planning Risks
4. Budget / Financial Risks
5. Operational / Procedural Risks
6. Technical / Functional / Performance Risks
7. Other Unavoidable Risks</td><td>Any 4 correct risk types: 2 M</td></tr>
<tr><td></td><td>e)</td><td>Define empirical estimation approach.</td><td>2 M</td></tr>
<tr><td></td><td>Ans</td><td>Empirical estimation techniques are based on making an educated guess of the project parameters. While using this technique, prior experience with development of similar products is helpful. Although empirical estimation techniques are based on common sense, different activities involved in estimation have been formalized over the years. Two popular empirical estimation techniques are: Expert judgment technique and Delphi cost estimation.</td><td>Correct Definition: 2 M</td></tr>
<tr><td></td><td>f)</td><td>Differentiate between quality assurance and quality control. (Any two points)</td><td>2 M</td></tr>
<tr><td></td><td>Ans</td><td>
<table>
<tr><th colspan="2">Quality Assurance</th><th colspan="2">Quality Control</th></tr>
<tr><td>1.</td><td>It is a procedure that focuses</td><td>1.</td><td>It is a procedure that</td></tr>
</table>
</td><td>Any two correct differentiation points: 2 M</td></tr>
</table>

				on providing assurance that quality requested will be achieved.		focuses on fulfilling the quality requested.	
			2.	QA aims to prevent defects.	2.	QC aims to identify and fix defects.	
			3.	It is a method to manage the quality- verification.	3.	It is a method to verify the quality-validation.	
			4.	It does not involve executing the program.	4.	It always involves executing a program.	
			5.	It's a preventive technique.	5.	It's a corrective technique.	
			6.	It's a proactive measure.	6.	It's a reactive measure	
			7.	It is the procedure to create the deliverables	7.	It is the procedure to verify that deliverables.	
			8.	QA involves in full software development life cycle.	8.	QC involves in full software testing life cycle.	
			9.	QA defines standards and methodologies in order to meet the customer requirements.	9.	QC confirms that the standards are followed while working on the product.	
			10.	It is performed before Quality Control.	10.	It is performed only after QA activity is done.	
			11.	It is a low-level activity, which can identify an error and mistakes which QC cannot.	11.	It is a High-Level Activity, which can identify an error that QA cannot.	
			12.	Its main motive is to prevent defects in the system. It is less time-consuming activity.	12.	Its main motive is to identify defects or bugs in the system. It is a more time-consuming activity.	
			13.	QA ensures that everything is executed in the right way, and that is why it falls under	13.	QC ensures that whatever we have done is as per the requirement, and that is why it falls under	

				verification activity		validation activity.	
			14.	It requires the involvement of the whole team.	14.	It requires the involvement of the Testing team.	
			15.	The statistical technique applied on QA is known as SPC or Statistical Process Control (SPC).	15.	The statistical technique applied to QC is known as SQC or Statistical Quality Control.	
	g)	**Define Software Quality Assurance Plan (SQAP).**					**2 M**
	Ans	• Software quality assurance plan consists of the auditing and reporting functions of management. • The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goals.					Correct Definition: 2 M
2.		**Attempt any <u>THREE</u> of the following:**					**12 M**
	a)	**Explain characteristics of software.**					**4 M**
	Ans	1. Software is developed or engineered; it is not manufactured in the classical sense. • Although some similarities exist between software development and hardware manufacture, the two activities are fundamentally different. • In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are non-existent (or easily corrected) for software. • Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different. • Software costs are concentrated in engineering. This means that software projects cannot be managed as if they were manufacturing projects. 1. Software doesn't "wear out."					Correct characteristics: 4 M



The idealized curve as shown in above figure is an oversimplification of actual failure models for software.

However, the implication is clear software doesn't wear out.

But it does deteriorate!

- This contradiction can best be explained by considering the "actual curve" shown in Figure.
- During its life, software will undergo change (maintenance). As changes are made, it is likely that some new defects will be introduced, causing the failure rate curve to spike as shown in figure.
- Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again. Slowly, the minimum failure rate level begins to rise—the software is deteriorating due to change.

2. Although the industry is moving toward component-based construction, most software continues to be custom built.

   The reusable components have been created so that the engineer can concentrate on the truly innovative elements of a design, that is, the parts of the design that represent something new.

   - In the software world, it is something that has only begun to be achieved on a broad scale. A software component should be designed and implemented so that it can be reused in many different programs.
   - A software component should be designed and implemented so that it can be reused in many different programs. Modern reusable components encapsulate both data and the processing that is applied to the data, enabling the software engineer to create new applications from reusable parts.
   - For example, today's interactive user interfaces are built with reusable

components that enable the creation of graphics windows, pull-down menus, and a wide variety of interaction mechanisms.

3. A software component should be designed and implemented so that it can be reused in many different programs.
   - It is the responsibility of software engineer to design and implement a software component in such a way that it should be reused easily in many different programs.
   - Latest reusable components summarize both data as well as the processing, which is applied to the data, which helps the software engineer to develop new applications from existing components.

b) Describe the notations used for preparing a structured chart. **4 M**

Ans Notations used in construction of structured chart are: Any two correct notations with description: 4 M

1. Module

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

   a. Control Module

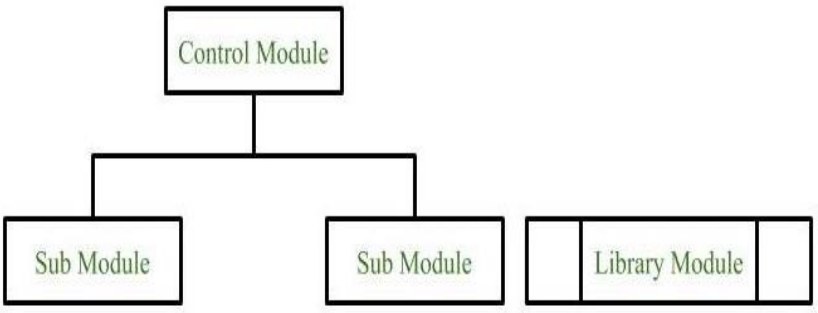
      A control module branches to more than one sub module.

   b. Sub Module

      Sub Module is a module which is the part (Child) of another module.

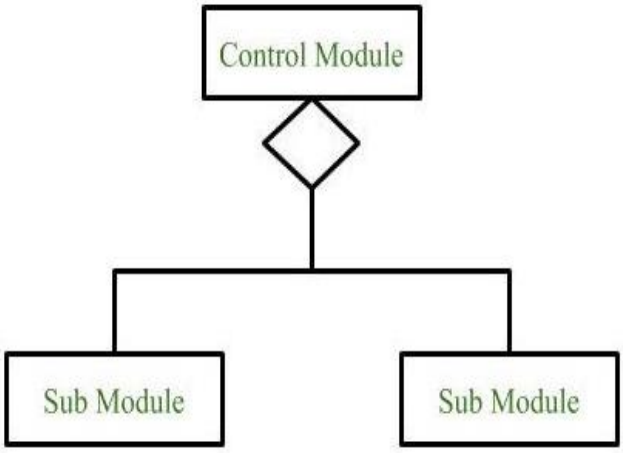
   c. Library Module

      Library Module are reusable and invokable from any module.



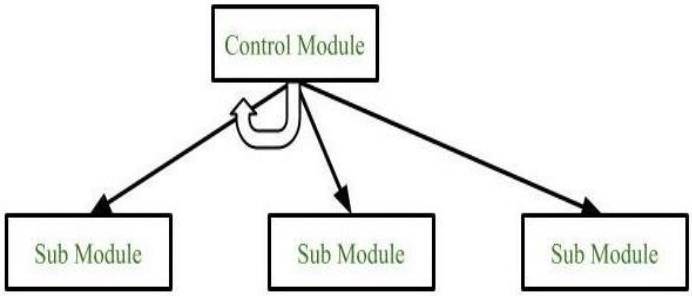
2. Conditional Call

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



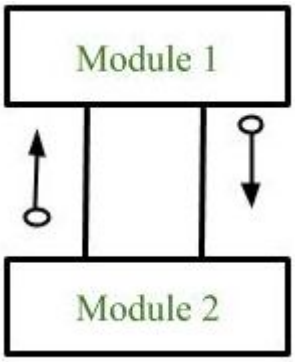
3. Loop (Repetitive call of module)

It represents the repetitive execution of module by the sub module.



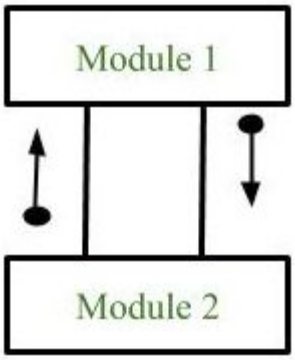
4. Data Flow

It represents the flow of data between the modules. It is represented by a 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 where all the information are to be stored.

	c)	**Explain the following 4P's management spectrum.**	**4 M**
	Ans	The management Spectrum: 4p's Effective software project management focuses on the four P's: People, Product, Process, and Project. The People: The "people factor" is so important that the Software Engineering Institute has developed a People Capability Maturity Model (People CMM) to continually improve its ability to attract, develop, motivate, organize, and retain the workforce needed to accomplish its strategic business objectives. 2. The people capability maturity model defines the following key practice areas for software people: a. Staffing b. communication and coordination c. work environment d. performance management e. Training, compensation, competency analysis and development, career development,	Explanation each element of management spectrum: 1 M

workgroup development, team/culture development and others.

3. Organizations that achieve high levels of People-CMM maturity have higher likelihood of implementing effective software project management practices.

The Product:

1. Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered and technical and management constraints should be identified.

2. Without this information, it is impossible to define reasonable (and accurate) estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule that provides a meaningful indication of progress.

3. Objectives identify the overall goals for the product (from the stakeholders' points of view) without considering how these goals will be achieved.

4. Scope identifies the primary data, functions, and behaviors that characterize the product.

5. The alternatives enable managers and practitioners to select a "best" approach, given the constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces, and other factors.

The Process:

1. A software process provides the framework from which a comprehensive plan for software development can be established.

2. A small number of framework activities are applicable to all software projects, regardless of their size or complexity.

3. Number of different task 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.

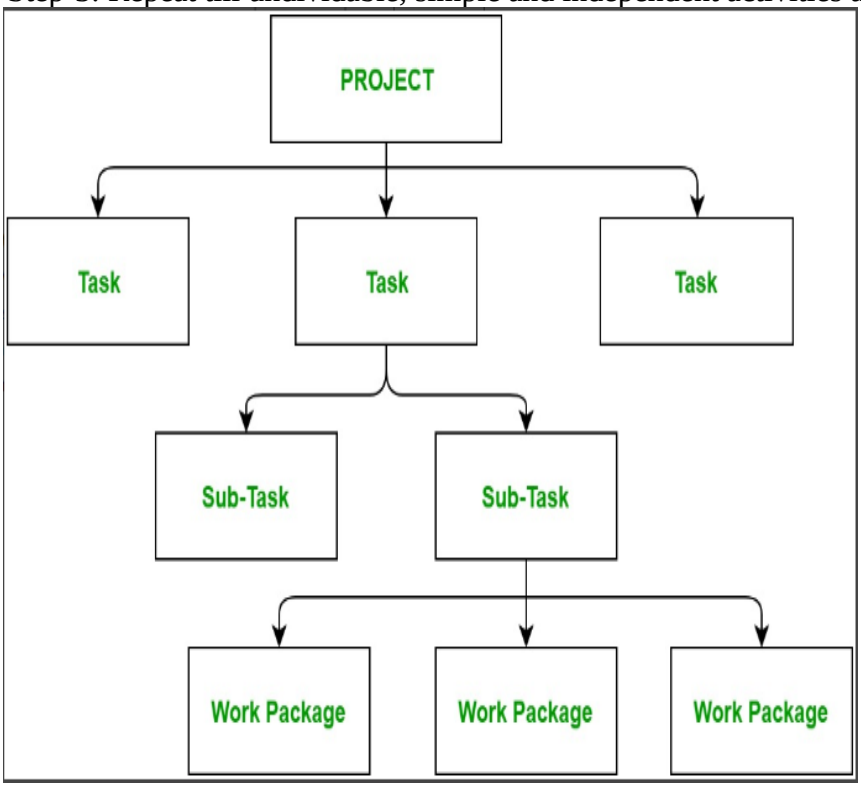
4. Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement occur throughout the process.

The Project:

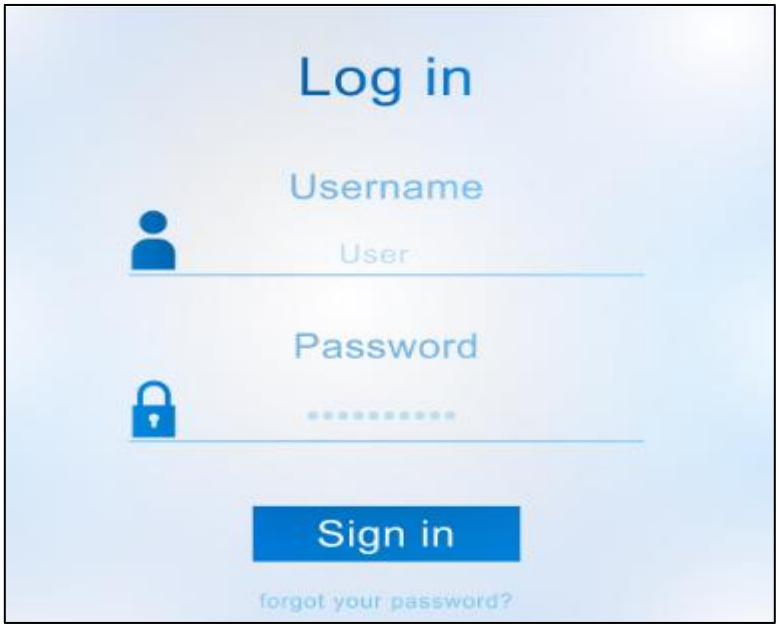
1. To manage complexity, we conduct planned and controlled software projects.

2. The success rate for present-day software projects may have improved but our project failure rate remains much higher than it should be.

3. To avoid project failure, a software project manager and the software engineers who

		build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a common-sense approach for planning, monitoring, and controlling the project.	
	d)	**Describe work breakdown structure with diagram.**	**4 M**
	Ans	Dividing complex projects into simpler and manageable tasks is the process identified as Work Breakdown Structure (WBS). Usually, the project managers use this method for simplifying the project execution. In WBS, much larger tasks are broken down to manageable chunks of work. These chunks can be easily supervised and estimated. WBS is not restricted to a specific field when it comes to application. This methodology can be used for any type of project management. Steps: • Step-1: Identify the major activities of the project. • Step-2: Identify the sub-activities of the major activities. • Step-3: Repeat till undividable, simple and independent activities are created.  <u>Construction of a WBS</u> Identifying the main deliverables of a project is the starting point for deriving a work breakdown structure. This important step is usually done by the project managers and the subject matter experts (SMEs) involved in the project. Once this step is completed, the subject matter experts start breaking down the high-level tasks into smaller chunks of work. In the process of breaking down the tasks, one can break them down into different levels of detail. One can detail a high-level task into ten sub-tasks while another can detail the same high-level task into 20 sub-tasks. Therefore, there is no hard and fast rule on how you should breakdown a task in WBS.	Correct explanation with diagram: 4 M

		Rather, the level of breakdown is a matter of the project type and the management style followed for the project. In general, there are a few "rules" used for determining the smallest task chunk. In "two weeks" rule, nothing is broken down smaller than two weeks' worth of work. This means, the smallest task of the WBS is at least two-week long. 8/80 is another rule used when creating a WBS. This rule implies that no task should be smaller than 8 hours of work and should not be larger than 80 hours of work. One can use many forms to display their WBS. Some use tree structure to illustrate the WBS, while others use lists and tables. Outlining is one of the easiest ways of representing a WBS.	
3.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**State the requirements to apply RAD.**	**4 M**
	Ans	• Following are the requirements to apply RAD model: • RAD model can be applied successfully to the projects in which clear modularization is possible. • It should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months. • Rapid Application Development is best when you've got a tight deadline to meet or are under pressure to deliver something that works. • RAD works well only if high skilled engineers (developers / designers) are available and the customer is also committed to achieve the targeted prototype in the given time frame. • RAD model should be chosen only if domain experts are available with relevant business knowledge. • RAD works well wherever there's a greater focus on user interface rather than non-GUI programs. • RAD should be used only when a system can be modularized to be delivered in an incremental manner. • It should be used only if the budget permits use of automated code generating tools.	Relevant requirement explanation give marks
	b)	**Describe any four software coding principles.**	**4 M**
	Ans	**The coding principles are that guide the coding task and are closely aligned with programming style , programming language , and programming methods. Coding principles can be stated as : -** **Preparation principles: Before you write one line of code, be sure you** • Understand of the problem you're trying to solve. • Understand basic design principles and concepts. • Pick a programming language that meets the needs of the software to be built and the environment in which it will operate. • Select a programming environment that provides tools that will make your work easier. • Create a set of unit tests that will be applied once the component you code is	For each principle give 1 M. Explain any 4 principles

		completed. **Programming principles: As you begin writing code, be sure you.** •Constrain your algorithms by structured programming practice. • Consider the use of pair programming. • Select data structures that will meet the needs of the design. • Understand the software architecture and create interfaces that are consistent with it. • Keep conditional logic as simple as possible. • Create nested loops in a way that makes them easily testable. • Select meaningful variable names and follow other local coding standards. · Write code that is self-documenting. • Create a visual layout (e.g., indentation and blank lines) that aids understanding. **Validation Principles: After you've completed your first coding pass, be sure you** • Conduct a code walkthrough when appropriate. • Perform unit tests and correct errors you've uncovered. • Refactor the code.	
	c)	**Prepare decision table for accessing secured network.**	**4 M**
	Ans	• Decision table is a software testing technique used to test system behavior for different input combinations. • This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form. That is why it is also called as a **Cause-Effect table** where Cause and effects are captured for better test coverage.  Fig: Login Screen of Network • The condition is simple if the user provides correct username and password the	For correct decision table with proper condition give 3 M and explanation 1 M

user will be redirected to the homepage.

- If any of the input is wrong, an error message will be displayed.

Decision Base Table for Login Screen

Decision Table

Conditions	Rule 1	Rule2	Rule3	Rule 4
Username(T/F)	F	T	F	T
Password(T/F)	F	F	T	T
Output(E/H)	E	E	E	H

Legend:

T – Correct username/password

F – Wrong username/password

E – Error message is displayed

H – Home screen is displayed

Interpretation:

- Case 1 – Username and password both were wrong. The user is shown an error message.
- Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
- Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
- Case 4 – Username and password both were correct, and the user navigated to homepage.

	d)	**State project size estimation techniques and explain any one.**	**4 M**
	Ans	Currently two metrics are popularly being used widely to estimate size: Lines of code (LOC) and function point (FP). **a) Lines of Code (LOC)** • LOC is the simplest among all metrics available to estimate project size. • This metric is very popular because it is the simplest to use. • Using this metric, the project size is estimated by counting the number of source instructions in the developed program. • Obviously, while counting the number of source instructions, lines used for commenting the code and the header lines should be ignored. • Accurate estimation of the LOC count at the beginning of a project is	For listing project size estimation technique 1M, 3 M for explanation of any one technique

very difficult.

- To estimate the LOC count at the beginning of a project, project managers usually divide the problem into modules, and each module into submodules and so on, until the sizes of the different leaf-level modules can be approximately predicted.
- To be able to do this, experience in developing similar products is helpful. By using the estimation of the lowest level modules, project managers arrive at the total size estimation.

Consider example of mechanical CAD software.

The mechanical CAD software will accept two dimensional and three-dimensional data. User will interact with system through user interface and peripheral devices like mouse, plotter, laser printer. All geometric data and supporting information are stored in database. Design analysis modules will be developed to produce the required output, which will be displayed on variety of graphic devices.

For size estimation CAD software is divided into major functions and size estimation of each function is done.

Function	Estimated LOC
User interface	2,300
Two-dimension geometric analysis	5,300
Three-dimension geometric analysis	6,800
Database management	3,500
Computer Graphics display facilities	4,950
Peripheral Control Function	2,100
Design analysis module	8,400
Estimated line of code	33,200

b) Function Point (FP):

- The conceptual idea behind the function point metric is that the size of a software product is directly dependent on the number of different Functions or features it supports.
- A software product supporting many features would certainly be of larger size than a product with a less number of features.
- Each function when invoked reads some input data and transforms it to the corresponding output data.
- For example, the issue book feature of a Library Automation Software

		takes the name of the book as input and displays its location and the number of copies available. • Thus, a computation of the number of input and the output data values to a system gives some indication of the number of functions supported by the system. • In addition to the number of basic functions that a software performs, the size is also dependent on the number of files and the number of interfaces. • Besides using the number of input and output data values, function point metric computes the size of a software product (in units of functions points or FPs) using three other characteristics of the product as shown in the following expression. • The size of a product in function points (FP) can be expressed as the weighted sum of these five problem characteristics. • The weights associated with the five characteristics were proposed empirically and validated by the observations over many projects. • Function point is computed in two steps. The first step is to compute the unadjusted function point (UFP). UFP = (Number of inputs)*4 + (Number of outputs)*5 + (Number of inquiries)*4 + (Number of files)*10 + (Number of interfaces)*10 • Once the unadjusted function point (UFP) is computed, the technical complexity factor (TCF) is computed next. • TCF refines the UFP measure by considering fourteen other factors such as high transaction rates, throughput, and response time requirements, etc. • Each of these 14 factors is assigned from 0 (not present or no influence) to 6 (strong influence). • The resulting numbers are summed, yielding the total degree of influence (DI). • Now, TCF is computed as TCF=(0.65+0.01*DI). As DI can vary from 0 to 70, TCF can vary from 0.65 to 1.35. Finally, FP=UFP*TCF.	
4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Explain adaptive software development method w.r.to** **i) Speculation.** **ii) Collaboration.**	**4 M**
	Ans	**i) Speculation** • During *speculation*, the project is initiated, and *adaptive cycle planning* is	For Speculation

		conducted. • Adaptive cycle planning uses project initiation information—the customer's mission statement, project constraints (e.g., delivery dates or user descriptions), and basic requirements—to define the set of release cycles (software increments) that will be required for the project. • No matter how complete and farsighted the cycle plan, it will invariably change. • Based on information obtained at the completion of the first cycle, the plan is reviewed and adjusted so that planned work better fits the reality in which an ASD team is working. **ii)Collaboration** • Motivated people use *collaboration* in a way that multiplies their talent and creative output beyond their absolute numbers. • This approach is a recurring theme in all agile methods. But collaboration is not easy. • It encompasses communication and teamwork, but it also emphasizes individualism, because individual creativity plays an important role in collaborative thinking. • It is, above all, a matter of trust. • People working together must trust one another to (1) criticize without animosity, (2) assist without resentment, (3) work as hard as or harder than they do, (4) have the skill set to contribute to the work at hand, and (5) Communicate problems or concerns in a way that leads to effective action.	Explanation 2 M and for collaboration Explanation 2 M
	b)	**Describe any four core principles of software engineering practices.**	**4 M**
	Ans	Following are core principles of Software Engineering Practices: **1. Reason it all exists.** i. The software system exists to provide value for the user. ii. Before specifying the problem the requirement and the specifications have to be laid down. iii. The hardware and the software platform to be decided for implementation. **2. Keep it simple stupid** i. The terms and the design used for development of the project should be kept simple and easily understandable. ii. All the terms used should be easy to facilitate the basic concept of the project. **3. Maintain the vision** i. A clear vision is important for the development of software. ii. Compromising the architectural vision of the project weakens the development of the software.	For each principle give 1 M

		iii. The developer should hold the vision and ensure the successful development and deployment of the software. **4. What you reproduce, someone else will have to consume.** (Implement knowing someone else will have to understand what you are doing) i. Always specify, design, and implement knowing that someone else is going to understand what is being developed. ii. Customers for the product development are very large. iii. Design the data structure and the implementation keeping implementation in mind and the end user. iv. Code with the concern that the product must be implemented and maintained by the end user. **5. Be open to the future** i. The system designed today should be adaptable to the development and changes in the future at a low cost. ii. There should not be many changes to the software to adapt to the new changes in the future development. **6. Plan ahead for reuse** i. The design and specifications should be developed in such a way that they can be reused for other implementations. ii. The code and the design should be well documented for the use in future. **7. Think!** i. Before designing and implementation a proper thought should be to the result. ii. Proper data structure and the design and implementation strategy should be developed if the software needs modification in the future.	
	c)	**Describe RMMM strategy.**	**4 M**
	Ans	Risk mitigation, monitoring, and management (RMMM) plan. A risk management strategy can be included in the software project plan, or the risk management steps can be organized into a separate Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan. Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. Risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives: (1) To assess whether predicted risks do, in fact, occur. (2) To ensure that risk aversion steps defined for the risk are being properly applied;	Correct explanation give marks

and

(3) To collect information that can be used for future risk analysis.

In many cases, the problems that occur during a project can be traced to more than one risk.

Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout the project).

An effective strategy must consider three issues: • Risk avoidance • Risk monitoring • Risk management and contingency planning.

If a software team adopts a proactive approach to risk, avoidance is always the best strategy.

This is achieved by developing a plan for risk mitigation.

Consider example of Staff Turnover risk:

To mitigate this risk, project management must develop a strategy for reducing turnover.

Among the possible steps to be taken are

• Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market).

• Mitigate those causes that are under our control before the project starts.

• Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.

• Organize project teams so that information about each development activity is widely dispersed.

• Define documentation standards and establish mechanisms to be sure that documents are developed in a timely manner.

• Conduct peer reviews of all work (so that more than one person is "up to speed).

• Assign a backup staff member for every critical technologist. As the project proceeds, risk monitoring activities commence.

The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely.

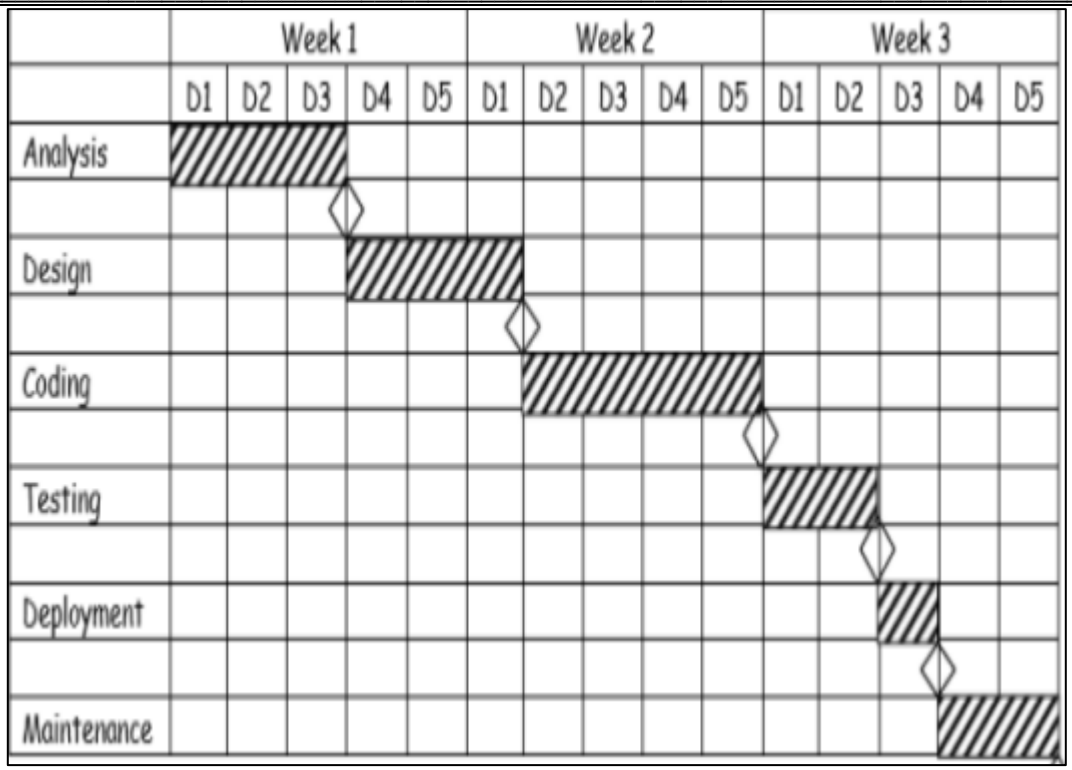
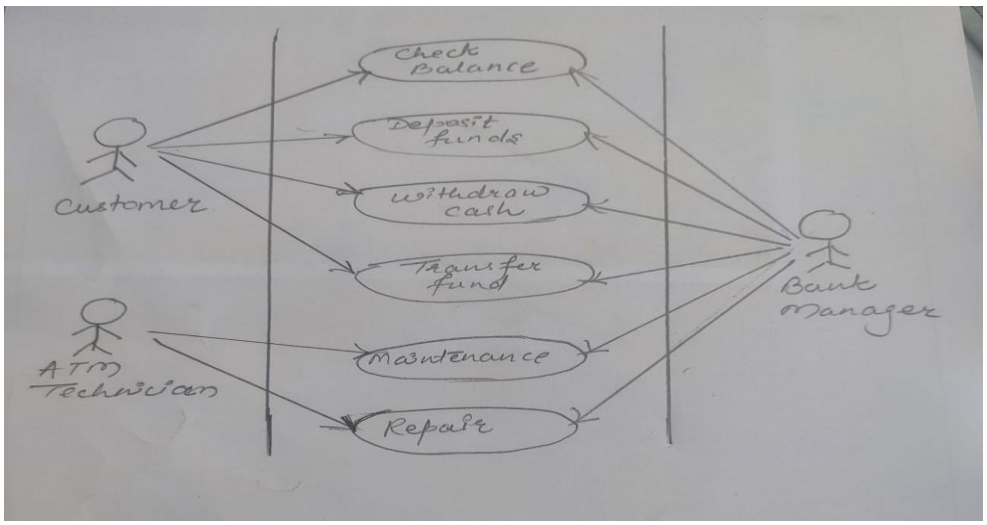
In the case of high staff turnover, the following factors can be monitored:

• General attitude of team members based on project pressures.

• The degree to which the team has jelled.

• Interpersonal relationships among team members.

		• Potential problems with compensation and benefits. • The availability of jobs within the company and outside it. In addition to monitoring these factors, the project manager should monitor the effectiveness of risk mitigation steps. RMMM steps incur additional project cost. Part of risk management, therefore, is to evaluate when the benefits accrued by the RMMM steps are outweighed by the costs associated with implementing them. Project planner performs a classic cost/benefit analysis.	
	d)	**Explain Heuristic method of cost estimation approach.**	**4 M**
	Ans	Heuristic techniques assume that the relationships among the different project parameters can be modeled using suitable mathematical expressions. Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting the value of the basic parameters in the mathematical expression. **Different heuristic estimation models can be divided into the following two classes: single variable model and the multi variable model.** **Single variable estimation models** provide a means to estimate the desired characteristics of a problem, using some previously estimated basic (independent) characteristic of the software product such as its size. A single variable estimation model takes the following form: Estimated Parameter = c1 * e1d1 In the above expression, e is the characteristic of the software which has already been estimated (independent variable). Estimated Parameter is the dependent parameter to be estimated. The dependent parameter to be estimated could be effort, project duration, staff size, etc. c1 and d1 are constants. The values of the constants c1 and d1 are usually determined using data collected from past projects (historical data). The basic COCOMO model is an example of single variable cost estimation model. **A multivariable cost estimation model takes the following form:** Estimated Resource = c1 * e1d1 + c 2 * e2 d2 + ... Where e1, e2, … are the basic (independent) characteristics of the software already estimated, and c1, c2, d1, d2, … are constants.	2 M for Single variable estimation model and 2 M for multi variable estimation model
	e)	**Prepare Gantt chart for hostel management system.** **(Five days a week). Consider phases of SDLC.**	**4 M**

	Ans		Correct timeline chart 4 M
5.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Draw use-case diagram for ATM system with minimum four use cases and two actors.**	**6 M**
	Ans	Use case diagram for ATM system with minimum four use cases and two actors 	Use case diagram, 2 actors : 2 M : 1 M for each Four use cases: 4 M: 1 M each: (total 6 M) Any other relevant use cases and actors shall be given marks.
	b)	**Differentiate between black box testing and white box testing (any six points).**	**6 M**
	Ans	Differences between black box testing and white box testing	Any valid 6

S. No.	Black Box Testing	White Box Testing
1.	It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
2.	Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
3.	It is mostly done by software testers.	It is mostly done by software developers.
4.	It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
5.	It is a functional test of the software.	It is a structural test of the software.
6.	This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detail design document.
7.	It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
8.	It is also called closed box testing.	It is also called as clear box testing.
9.	It is least time consuming.	It is most time consuming.

differences between black box testing and white box testing: 6 M, 1 M each) Any more relevant points shall be given marks.

		10.	**Types of Black Box Testing:** • Functional Testing • Non-functional testing • Regression Testing	**Types of White Box Testing:** • Path Testing • Loop Testing • Condition testing	
		11.	It is less exhaustive as compared to white box testing.	It is comparatively more exhaustive than black box testing.	
	c)	**Use COCOMO model to calculate** **i) Effort** **ii) Development time** **if estimated size of project is 500 KLOC using organic,** **Semi-detached and Embedded mode.**			**6 M**
	Ans				Effort : all 3 modes :3 M : 1 M each Development time : all 3 modes :3 M : 1 M each = total 6 M

Project	a_b	b_b	c_b	d_b
Organic mode	2.4	1.05	2.5	0.38
Semidetached mode	3.0	1.12	2.5	0.35
Embedded mode	3.6	1.20	2.5	0.32

Effort ,E = a_b (KLOC)$^{b}{}_{b}$ persons-months

Development time D = c_b (E)$^{d}{}_{b}$ months

In organic mode :

i) E= 2.4 * (500) $^{1.05}$

= 2.4 * 682.21

=1637.30 person-months

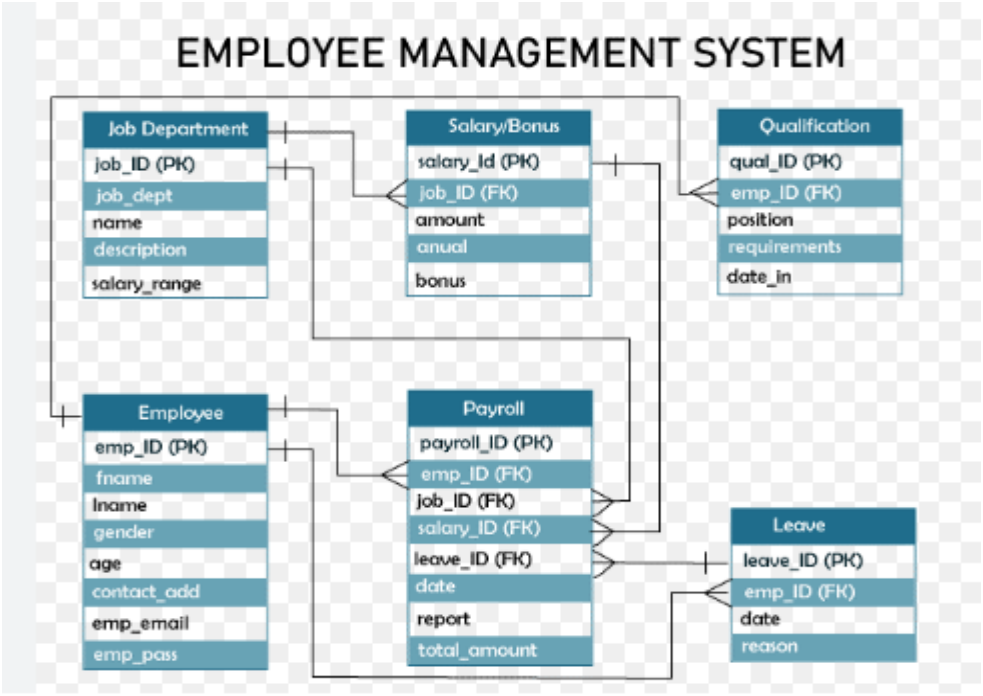
ii) D = 2.5 * (1637.30) $^{0.38}$

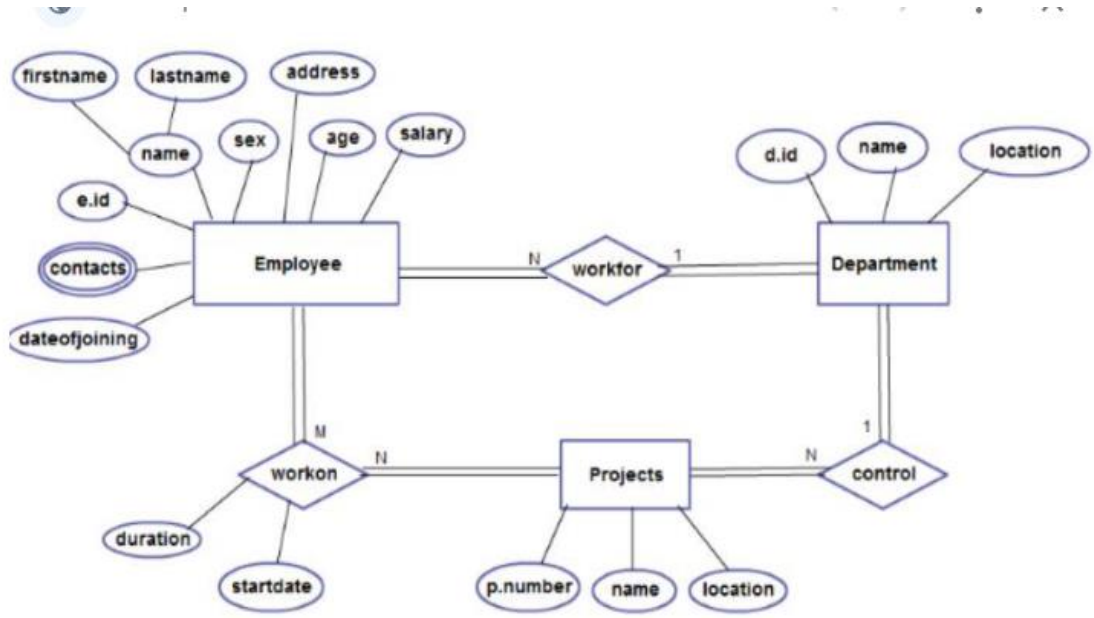
=2.5 *16.64

=41.6 months

In Embedded mode :

i) E= 3.6*(500) $^{1.20}$

		= 3.6 * 1732.86 =6238.29 Person-Months ii) D= 2.5 * (6238.29) $^{0.32}$ =2.5 * 16.38 = 40.95 Months **In Semidetached mode :** i) E= 3.0 *(500) $^{1.12}$ = 3.0 * 1054.01 =3162.04 Person-Months ii) D= 2.5 * (3162.04) $^{0.35}$ =2.5 * 16.78 = 41.95 Months	
6.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Draw and explain conceptual data model with E-R diagram for employee management system.**	**6 M**
	Ans	DATA MODEL  	Only Data model done – 3 marks Only ER dia drawn – 3 marks (Number of entities, attributes identified may differ. Consider any relevant terms)

		E-R DIAGRAM 	
	b)	**Describe six sigma and state the phases of DMAIC and DMADV.**	**6 M**
	Ans	Six sigma : 1. Six sigma is the process of producing high and improved quality output. 2. This can be done in two phases – identification and elimination. The cause of defects is identified and appropriate elimination is done which reduces variation in whole processes. 3. A six sigma method is one in which 99.99966 percentage of all the products to be produced have the same features and are of free from defects. 4. The Characteristics of Six Sigma are as follows: (a) Statistical Quality Control: Standard Deviation in statistics is used for measuring the quality of output. (b) Methodical Approach:-The Six Sigma is not a merely quality improvement strategy in theory, as it features a well-defined systematic approach of application in DMAIC and DMADV which can be used to improve the quality of production. DMAIC is an acronym for Design-Measure- Analyze-Improve-Control. The alternative method DMADV stands for Design-Measure- Analyze-Design-Verify. (c) Fact and Data-Based Approach:-The statistical and methodical as pect of Six Sigma shows the scientific basis of the technique. (d) Project and Objective-Based Focus:- The Six Sigma process is implemented for an organization's project tailored to its specification and requirements. (e) Customer Focus:- The customer focus is fundamental to the Six Sigma	Description of six sigma : 3 M ; DMAIC phases -1.5 M and DMADV phases -1.5 M

		approach. The quality improvement and control standards are based on specific customer requirements. (f) Teamwork Approach to Quality Management: - The Six Sigma process requires organizations to get organized when it comes to controlling and improving quality. DMADV PHASES THE PHASES OF DMADV ARE: 1. **Define:** It covers the process mapping and flow-charting, project charter development, problem-solving tools, and so-called 7-M tools. 2. **Measure:** It includes the principles of measurement, continuous and discrete data, and scales of measurement, an overview of the principle of variations and repeatability and reproducibility (RR) studies for continuous and discrete data. 3. **Analyze:** It covers establishing a process baseline, how to determine process improvement goals, knowledge discovery, including descriptive and exploratory data analysis and data mining tools, the basic principle of Statistical Process Control (SPC), specialized control charts, process capability analysis, correlation and regression analysis, analysis of categorical data, and non-parametric statistical methods. 4. **Improve:** It covers project management, risk assessment, process simulation, and design of experiments (DOE), robust design concepts, and process optimization. 5. **Control:** It covers process control planning, using SPC for operational control and PRE-Control. DMAIC PHASES ARE: 1. **Define:** It defines the problem or project goal that needs to be addressed. 2. **Measure:** It measures and determines the customer's needs and specifications. 3. **Analyze:** It analyzes the process to meet customer needs. 4. **Design:** It can design a process that will meet customer needs. 5. **Verify:** It can verify the design performance and ability to meet customer needs.	
	c)	**State requirements for given modules of online shopping system.** **i) Order module** **ii) Accountant module** **iii) Categories module**	**6 M**

	Ans	This is with perspective of online shopping system. Requirements for following modules will be as **i) Order module :** a. Getting name of Item b. Getting the Item id c. Getting information of item price d. Getting information of quantity of item e. Information of availability of Item **ii) Accountant module** a. Getting the information of list of items purchased b. Bill generation c. Bill calculation d. Getting information of item price e. Generating the bill identification numbers **iii) Categories module** a. Getting information of number of categories b. Information of sub-categories of product c. Getting information on brands of various categories d. Information about the sizes in categories	Requirements of 3 modules : 2 M each; total 6 M (any other relevant requirements shall be given marks)