Reg. No.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**SRM Institute of Science and Technology**

**SET A**

**College of Engineering and Engineering and Technology**

**School of Computing**

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

**Academic Year: 2022-23 EVEN**

**Test: CLAT 3**  **Date: 10 May 2023**

**Course Code & Title: 18CSC206J** Software Engineering and Project Management **Duration:** 100 mins

**Year & Sem:** II Year / IV Sem. **Max. Marks:** 50

**Course Articulation Matrix: *(to be placed)***

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Outcomes** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **P10** | **P11** | **P12** |
| **CO1** | H | M | L |  |  |  |  |  | L | L | H | M |
| **CO2** | H | M | M | M | H |  |  |  | L | L | L | M |
| **CO3** | H |  | M |  | H |  |  |  | H | L | L | M |
| **CO4** |  | M |  |  | M |  |  | M | H | M | L |  |
| **CO5** |  |  |  |  |  | M | L | L |  | H | M |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Part – A (1 x 10 = 10 Marks) Answer all** | | | | | | |
| **Q. No** | **Question** | **Marks** | **BL** | **CO** | **PO** | **PI Code** |
| **1** | To test the application under test, what must be prepared to match an environment that is close to the environment under which the proposed application will be deployed for production?   1. Test plan 2. Test Execution plan 3. **Test bed** 4. Test case | 1 | 1 | 4 | 2 | 2.1.2 |
| **2** | What are the ways to measure the performance of the testing team?   1. Defect fix 2. Test execution 3. **Defect count per hour** 4. Test case design | 1 | 1 | 4 | 3 | 2.1.1 |
| **3** | Which of the following is a form of functional testing?   1. Security level testing 2. **Boundary value analysis** 3. Performance testing 4. Usability testing | 1 | 1 | 4 | 1 | 2.1.1 |
| **4** | When an expected result is not specified in test case template then \_\_\_\_\_\_\_\_\_\_\_.   1. We cannot run the test. 2. It may be difficult to repeat the test. 3. **It may be difficult to determine if the test has passed or failed.** 4. We cannot automate the user inputs. | 1 | 2 | 4 | 4 | 2.1.1 |
| **5** | Which is not true in case of Unit Testing?   1. **It decreases the software development speed.** 2. It can’t be expected to catch every error in a program. 3. In this tester evaluates if individual units of source code are fit for use. 4. It is usually conducted by the development team. | 1 | 2 | 4 | 3 | 2.3.2 |
| **6** | The modification of the software to match changes in the ever changing  environment, falls under which category of software maintenance?   1. Corrective 2. **Adaptive** 3. Perfective 4. Preventive | 1 | 1 | 5 | 1 | 2.1.1 |
| **7** | Which process is used to achieve system specification by thoroughly analyzing, understanding the existing system?   1. Program Restructuring 2. **Reverse Engineering** 3. Forward Engineering 4. None of the above | 1 | 1 | 5 | 3 | 2.1.2 |
| **8** | A \_\_\_\_\_\_\_\_\_\_is a report describes High-Quality Work Product /Evaluation Formal report that outlines the outcomes of all testing   1. Test Closure Report 2. Test Management Report 3. **Test Summary Report** 4. Test Cycle Report | 1 | 1 | 5 | 2 | 2.1.1 |
| **9** | The maintenance phase when there are significant changes in the software is:   1. Evolution 2. **Service** 3. Phase-out 4. Development | 1 | 2 | 5 | 3 | 2.1.1 |
| **10** | A pre-release of software that is given out to a  large group of users to try under real conditions   1. Alpha Release 2. **Beta Release** 3. Internal Release 4. External Release | 1 | 2 | 5 | 3 | 2.1.2 |
|  | | | | | | |
| Reg. No.   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |   **SRM Institute of Science and Technology**  **SET A**  **b**  **College of Engineering and Engineering and Technology**  **School of Computing**  SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu  **Academic Year: 2022-23 EVEN**  **Test: CLAT 3**  **Date: 10 May 2023**  **Course Code & Title: 18CSC206J** Software Engineering and Project Management **Duration:** 100 mins  **Year & Sem:** II Year / IV Sem. **Max. Marks:** 50  **Part – B**  **(4 x 4 =16 Marks) Answer any 4** | | | | | | |
| **Q. No** | **Question** | **Marks** | **BL** | **CO** | **PO** | **PI Code** |
| **11** | Discuss on Verification and Validation.  Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.  Verification means Are we building the product right?  Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing. | 4 | 1 | 4 | 2 | 2.5.2 |
| **12** | With a block diagram describe about defect tracking in test project  monitoring and control. | 4 | 2 | 4 | 3 | 5.4.2 |
| **13** | Explain about Test Automation.  Software Test automation makes use of specialized tools to control the execution of tests and compares the actual results against the expected result. Usually, regression tests, which are repetitive actions, are automated.  Testing Tools not only helps us to perform regression tests but also helps us to automate data set up generation, product installation, GUI interaction, defect logging, etc. Automation tools are used for both Functional and Non-Functional testing.  Criteria for Tool Selection:  For automating any application, the following parameters should be considered:  Data driven capabilities  Debugging and logging capabilities  Platform independence  Extensibility & Customizability  E-mail Notifications  Version control friendly  Support unattended test runs | 4 | 1 | 4 | 2 | 2.5.2 |
| **14** | Categorize the types of product release. Before release there are certain tasks to be completed enumerate the same with detailed description | 4 | 1 | 5 | 9 | 9.5.3 |
| **15** | Illustrate the different software Maintenance Techniques and their significance | 4 | 2 | 5 | 4 | 4.5.1 |
| **Part – C (2 x 12 =24 Marks)**  **Answer all** | | | | | | |
| **16.a.** | As a software tester, you have been asked to validate a software release for a client. Explain the various types of validation techniques that you would apply to ensure that the software release meets the client’s requirements.     * Validation testing is also known as dynamic testing as, in this case, the source code is actually run to determine that it is running per specifications. * During validation, unit, integration, system, and finally user acceptance testing are performed. * Unit testing is done to ensure each unit piece of source code is free from defects. * Once unit testing is done, then this piece of code is integrated with the main source code build. * But before integrating to the main build, it is strongly advisable to do local integration testing on the developer’s own computer. * Only when the source code runs smoothly and all integration tests pass, the source code should be integrated with the main build. * When all source code is thus integrated, the main build is ready for system testing. * All system tests are then performed and defects are fixed. * When the system testing is over and in fact the software product is shipped to customers, they do user acceptance testing. | 12 | 3 | 4 | 3 | 5.4.2 |
| **or** | | | | | | |
| **16.b.** | You have been tasked with validating the compatibility of a software application across multiple platforms and devices Describe the steps you would take to ensure that the application works seamlessly across different platforms and devices.   * Software testing is a vast field in itself, and so the common practice is to consider it as a separate project. * In those cases, it is known as an independent verification and validation project. * As such, a separate project plan is made for that project and is linked to the parent software development project. * There are many techniques available to execute software test projects. * It depends on the kind of test project. However, most test projects must have a test plan and a test strategy before the project can be ready for execution. * Often due to time constraints, testing cycles are cut short by project managers. * This leads to a half-tested product that is pushed out of the door. * In such cases, a large number of product defects are left undetected. * Ultimately, end users discover these defects. Fixing these defects at this stage is costly. * Moreover, they cannot be fixed one at a time. * They are to be taken in batches and are incorporated in maintenance project plans. * This leads to excessive costs in maintaining the software. * It is a lot cheaper to trap those bugs during the testing cycle and fix them. * It is appropriately said that “testing costs money but not testing, costs more!” * Test strategies should include things like test prioritization, automation strategy, risk analysis, etc. * Test planning should include a work breakdown structure, requirement review, resource allocation, effort estimation, tools selection, setting up communication channels, etc.   **Test Prioritization**   * Even before the test effort actually starts, it is of utmost importance that the test prioritization should be made. * First of all, all parts of the software product will not be used by end users with the same intensity. * Some parts of the product are used by end users extensively, while other parts are seldom used. * So the extensively used parts of the product should not have any defects at all and thus they need to be tested thoroughly.   **Test Prioritization**   * For making such a strategy, you must prioritize your testing. * Put a high priority on tests which are to be done for these critical parts of the software product and put a low priority on uncritical parts. * Then test the high priority areas first. * Once testing is thoroughly done for these parts, then you should start testing the low priority areas.   **Risk Management**   * The test manager should also do plan for all known risks that could impact the test project. * If proper risk mitigation planning is not done and a mishap occurs, then the test project schedule could be jeopardized, costs could escalate and/or quality could go down. * Some of the risks that can have severe, adverse impact on a test project include an unrealistic schedule, resource unavailability, skill unavailability, frequent requirement changes, etc. * Requirement changes pose a serious threat to testing effort because for each requirement change, the whole test plan gets changed. * The test team has to revise its schedule for additional work as well as to assess impact of the change on the test cases they have to recreate. * Some enthusiastic test engineers estimate much less effort than it actually should be. * In that case, the test manager would be in trouble trying to explain why testing is taking more than the scheduled time schedule. * In such cases, even after loading testing engineers more than 150%, the testing cycle get delayed. * This is a very common situation on most of the test projects. * This also happens because the marketing team agrees on unrealistic schedules with the customer in order to bag the project. * Even the test manager at that time feels that somehow he will manage it, but later on it proves impossible to achieve. * Other test engineers unnecessarily pad their estimate and later on, when the customer detects it, the test manager finds himself in a spot.   **Effort Estimation**   * For making scheduling, resource planning and budget for a test project, the test manager should make a good effort estimate. * Effort estimate should include information such as project size, productivity, and test strategy. * While project size and test strategy information comes after consultation with the customer, the productivity figure comes from experience and knowledge of the team members of the project team. * The wideband Delphi technique uses brainstorming sessions to arrive at effort estimate figures after discussing the project details with the project team.   **: Test Point Analysis**   * Productivity is derived from knowledge and experience of the test team members. * While productivity can be calculated objectively without taking reference from any statistical data, it makes sense to use past productivity data from previously executed projects to make productivity figures more realistic. * In case of iterative development, testing cycles will be short and iterative in nature. | 12 | 3 | 4 | 3 | 2.5.2 |
| **17.a.** | (i) Describe Software Maintenance Life Cycle (6 marks)  Phases of SMLC  (ii) Discuss on Boehm’s Model Osborne’s Model (6 marks)   * *Boehm’s model*: Boehm’s model is based on economic models and often involves calculating ROI (return of investment),for any planned maintenance. If ROI turns out to be good, then it is carried out or else it is dropped. * *Osborne’s model*: Osborne realized that difficulties in carrying out maintenance work are due togaps in communication. He proposed four steps to prevent this situation. He stated that a maintenance plan should include all change requests in the form of maintenance requirements. A quality assurance plan should accompany the maintenance plan. Metrics should be developed to measure and assess quality of work carried out during maintenance. Finally, reviews should be held after maintenance work to assess quality of work done. | 12 | 2 | 5 | 3 | 2.3.1 |
| **or** | | | | | | |
| **17.b.** | Discuss the different Software Maintenance engineering techniques and their significance   * Software maintenance is of four types: corrective, adaptive, preventive, and perfective maintenance. * If the software has some defects, then it will take a corrective maintenance to rectify it. * If there are some changes in the operating environment of the software product, then the product can be made useful by doing adaptive maintenance. * If there is an insecurity that although the product is running fine in future we may have difficulty in using it, then preventive maintenance is employed.   **Corrective**   * Even after thorough reviews and testing, the software product contains many defects when it goes into production. * These defects are uncovered as users start using the application. * They are logged with the support staff and after a sizable number of errors are detected, the software vendor instructs his maintenance team to create a patch to rectify them. * The maintenance team then makes a plan and fixes those defects. * After application of the patch containing the fixes, the software starts running without these defects.   **Adaptive**   * The operating environment in which a software product runs in operation includes the hardware and software platform as well as the interfaces for human and other machine interactions. * If any of these change over time, it becomes difficult to run the software product. * In such cases, it becomes necessary to do adaptive maintenance so that the software product becomes reusable. * This kind of maintenance may involve changing the interface or porting the application to another hardware/ software platform.   **Perfective**   * This kind of maintenance is needed when there is a change in the business environment, and thereby users need additional/modified functionality in the software product to do their tasks. * A business workflow may have changed, a business transaction may have changed, or an altogether new business transaction was represented in the software product. * For all these kinds of requirements, a perfective maintenance may be needed.   **Preventive**   * Generally after a lapse of time, there are likely changes in business or operative environment, or there may be changes in hardware/software environment. * These changes are bound to occur and they affect the way the software product operates. * Many of these changes can be perceived in advance. * In such cases, preventive maintenance on the software product can make sure that the product will be useful even after these environmental changes occur. | 12 | 2 | 5 | 3 | 2.3.1 |