COURSEPACK (Fall 2023-24)

SCHEME

The scheme is an overview of work-integrated learning opportunities and gets students out into the real world.This will give what a course entails.

<table>
<tr><td>Course Title</td><td colspan="3">Software Testing and Quality Assurance</td><td>Course Type</td><td colspan="5">Theory</td></tr>
<tr><td>Course Code</td><td colspan="3">E2UC502T</td><td>Class</td><td colspan="5">B.Tech. Core & All specialization (Vth Sem.)</td></tr>
<tr><td rowspan="6">Instruction delivery</td><td>Activity</td><td>Credits</td><td>Weekly Hours</td><td colspan="3" rowspan="2">Total Number of Classes per Semester</td><td colspan="2" rowspan="2">Assessment in Weightage</td></tr>
<tr><td>Lecture</td><td>3</td><td>3</td></tr>
<tr><td>Tutorial</td><td>0</td><td>0</td><td rowspan="4">Theory</td><td rowspan="4">Tutorial</td><td rowspan="4">Self-study</td><td rowspan="4">CIE</td><td rowspan="4">SEE</td></tr>
<tr><td>Practical</td><td>0</td><td>0</td></tr>
<tr><td>Self-study</td><td>0</td><td>6</td></tr>
<tr><td>Total</td><td>3</td><td>9</td><td>45</td><td>0</td><td>90</td><td>50%</td><td>50%</td></tr>
<tr><td rowspan="3">Names Course Instructors</td><td colspan="2">Course Lead</td><td colspan="2">Dr. Pardeep Singh</td><td colspan="2">Course Coordinator</td><td colspan="2">Dr. Azath Mohamed Hussain</td></tr>
<tr><td colspan="4">Theory</td><td colspan="2"></td><td colspan="2">Practical</td></tr>
<tr><td colspan="4">Ruchi Sharma
Pardeep Singh
Swati Sharma
Azath Mohamed Hussain
Manikant Panthi
Gautam Kumar
Garima Pandey
Shwet Ketu
R. Sathiya Priya
Radha Rani
Ravinder Beniwal
Indervati
K. Suresh
Nidhi Agarwal
Nidhi Sharma
Pravesh
Soumalya Ghosh
John A.
R. Radhika
Yashwant Soni
Dhirendra Kumar Shukla</td><td colspan="2"></td><td colspan="2">COURSEPACK |</td></tr>
</table>

	Suman Devi Kirti		

COURSE OVERVIEW

"Software Testing and Quality Assurance" course is designed to prepare students for careers in the field of software development, where they play a crucial role in ensuring the quality and reliability of software products.

PREREQUISITE COURSE

Prerequisite course required	Yes(√)	No

If, yes please fill in the details

Prerequisite course code	Prerequisite course name
	Any programming Language , DBMS

COURSE OBJECTIVE

- To teach fundamental concepts, importance, and principles of software testing in the software development life cycle.
- Familiarize students with debugging techniques, integrated development environments (IDEs), and essential software development practices for efficient coding and debugging.

COURSE OUTCOMES(COs)

After the completion of the course, the student will be able to:

Course Outcomes	Upon successful completion of this course, the student will be able to:
E2UC502T.1.	Design and execute test cases for software applications.
E2UC502T.2.	Apply verification and validation techniques to test cases for ensuring full coverage of paths.
E2UC502T.3.	Apply debugging tools, techniques and guidelines to identify and resolve bugs.
E2UC502T.4.	Develop programs using test-driven software engineering approach to ensure software quality.
E2UC502T.5.	Utilize Junit and Selenium tools for automated testing of software codes to enhance coverage of test cases.

BLOOM'S LEVEL OF THE COURSE OUTCOMES

Bloom's taxonomy is a set of hierarchical models used for the classification of educational learning objectives into levels of complexity and specificity. The learning domains are cognitive, affective, and psychomotor.

COMPREHENSIVE

CO No.	Bloom's Taxonomy Level(BTL)					
	Remember (**KL1**)	Understand (**KL2**)	Apply (**KL3**)	Analyze (**KL4**)	Evaluate (**KL5**)	Create (**KL6**)
E2UC102C.1			√	√		
E2UC102C.2			√		√	
E2UC102C.3			√			
E2UC102C.4			√			
E2UC102C.5				√	√	

PROGRAM OUTCOMES (POs):

PO1	**Computing Science knowledge:** Apply the knowledge of mathematics, statistics, computing science and information science fundamentals to the solution of complex computer application problems.
PO2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex computing science problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and computer sciences**.**
PO3	**Design/development of solutions:** Design solutions for complex computing problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern computing science and IT tools including prediction and modeling to complex computing activities with an understanding of the limitations.
PO6	**IT specialist and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional computing science and information science practice.
PO7	**Environment and sustainability**: Understand the impact of the professional computing science solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the computing science practice.

PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the IT analyst community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding of the computing science and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs):

PSO1	Have the ability to work with emerging technologies in computing requisite to Industry 4.0.
PSO2	Demonstrate Engineering Practice learned through industry internship and research project to solve live problems in various domains.

COURSE ARTICULATIONMATRIX

The Course articulation matrix indicates the correlation between Course Outcomes and Program Outcomes and their expected strength of mapping in three levels (low,medium,and high).

CO/PO Mapping (1 / 2 / 3 indicates strength of correlation) 3 - Strong, 2 - Medium, 1 – Low														
	Programme Outcomes (POs)													
COs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**
E2UC502T.**1**	-	2	2	-	-	-	-	-	-	-	-	2	-	-
E2UC502T.**2**	2	2	2	-	2	-	-	-	-	-	-	2	-	-
E2UC502T.**3**	2	2	2	-	3	-	-	-	-	-	-	2	2	-
E2UC502T.**4**	2	2	2	-	2	-	-	-	-	-	-	2	2	-
E2UC502T.**5**	2	2	2	-	3	-	-	-	-	-	-	2	3	-
1-Low,2-Medium,3-High														

COURSE ASSESSMENT

The course assessment patterns are the assessment tools used both in formative and summative examinations.

Type of Course (T)	**CIE**			**Total Marks**		**Grand Total Marks**	**Weightage** (CIE-SEE)
	IA1	MTE	IA2	CIE	SEE		
Theory	25	50	25	100	100	200	50-50

***Rubric for the course-based project**

Type of Assessment Tools	Preliminary Project Plan	Technical Seminar 1	Technical Seminar 2	Viva-voce
Course-based Project Work				

PPP (Preliminary Project Plan): *The preliminary project plan (PPP) provides an initial, overview of the project and all of its known parameters. It outlines the project's objectives, relevance to the program, merit, and conformity to current industry/government policy, proposed methodology, and expected outcomes. It should also include any known constraints related to the time frame (Gantt Chart), budget, and, etc.*

COURSE CONTENT

THEORY

THEORY:

Introduction

Software program and its objective, Software development techniques, software Matrices, top-down verses bottom-up approach, modular and structures programming. A brief introduction about object oriented approach.

Importance of Software Testing

Software testing and its importance, software development life cycle verses software testing life cycle, Deliverables, version and error control, Verification and Validation

Testing Techniques and Strategy

Unit testing, Integration testing, System testing, Acceptance testing

White-Box testing: Flow Graph notation, Cyclomatic Complexity, Graph matrices, control structure. and loop testing.

Black-Box testing: Equivalence partitioning, Boundary Value Analysis

Building Test Cases and Plans

Format of test cases, Du, dc and other data paths, Test data selection, branch coverage, statement coverage, pre-condition and post-condition, Test schedule and check pointing, suitable exercises for creating test cases for each type of Testing techniques.

Quality Assurance and Standards

Basic software quality parameters and its metrics, Software Configuration Change and types of errors, Quality management models: ISO, CMM

Debugging Technique and Tools

Integrated development environment, debugging, tracing, data inspection, exception errors, code and data redundancy, Junit and Selenium tool.

LESSON PLAN FOR INTEGRATED COURSES of 3 CREDITS

FOR THEORY 15 weeks * 3 Hours = 45 Classes) (1credit = 1Lecture Hour)

L.No.	Topics for Delivery	Theory/ Tutorial	Skills	Competency
1	Software program and its objective	Theory	Apply various approaches of Software Building	CO1
2	Software development techniques	Theory		
3	Software Metrices	Theory		
4	Top-down verses bottom-up approach	Theory		
5	Modular and structures programming	Theory		
6	Object Oriented Approach	Theory		
7	Software testing and its importance	Theory	Apply testing to identify errors	
8	Software development life cycle verses software testing life cycle	Theory		
9	Deliverables, version	Theory		CO3
10	Error control	Theory		
11	Verification and Validation techniques	Theory	Use verification validation techniques for test cases	CO1, CO2
12	Unit Testing and Integration Testing,	Theory		
13	System and Acceptance Testing	Theory		
14	White-Box testing: : Flow Graph notation	Theory	Apply Graph Techniques for evaluating software complexity	
15	White-Box testing: Cyclomatic Complexity	Theory		
16	White-Box testing: Graph matrices, control structure	Theory		
17	White-Box testing: L oop testing.	Theory		
18	Black-Box testing: Equivalence partitioning	Theory	Develop equivalence classes of test cases	
19	Black-Box testing: Boundary Value Analysis	Theory		
20	Building Test Cases and Plans	Theory	Designing and analysis of test cases for code coverage	
21	Format of test cases, Du, dc and other data paths	Theory		
22	Test data selection	Theory		
23	Branch coverage and Statement coverage	Theory		
24	Pre-condition and post-condition	Theory		
25	Test schedule and check pointing	Theory		
26	Suitable Exercises for creating test cases for	Theory		

	each type of testing			
27	Quality Assurance and Standards introduction	Theory	Identify Software Quality Standards	CO4
28	Software quality parameters and its metrics	Theory		
29	Software Configuration Change	Theory	Update Software Versions to ensure quality parameters	
30	Types of errors	Theory		
31	Quality management models: ISO	Theory		
32	Quality management models: CMM	Theory		
33	Debugging Technique and Tools	Theory	Apply debugging techniques for tracing bugs	CO3, CO4
34	Integrated development environment	Theory		
35	Debugging	Theory		
36	Tracing	Theory		
37	Data inspection	Theory	Remove redundancy and optimize code	
38	Exception errors	Theory		
39	Code and data redundancy	Theory		
40	Junit Package Introduction	Theory	Apply Automated Testing for Software Programs	CO1, CO5
41	Creating Test Cases for Number/Strings using Junit	Theory		
42	Selenium Tool Introduction	Theory		
43	Performing automated testing using Selenium	Theory		
44	Practice Problems for Manual Testing	Theory	Perform Manual Testing of programs	
45	Manual Testing of Software Programs	Theory		

`

BIBLIOGRAPHY

Text Book

1. Desikan S, Ramesh G, "Software Testing", Pearson Education, 2008.
2. Yogesh Singh," Software Testing", Cambridge University Press, 2011.
3. Dustin E, "Effective Software Testing", Pearson Education, 2007.
4. Mathur A.P, "Fundamentals of Software Testing", Pearson Education, 2008.

Reference Books

1. R. Pressman, "Software Engineering", 6th Edition, Tata McGraw-Hill.
2. Brian Marick, "The Craft of Software Testing", Pearson Education, 2008.
3. Rajani & Oak, "Software Testing : Methodology, Tools and Processes" TataMcGraw-Hill, 2007.
4. Robert Charles Metzger, "Debugging by Thinking: A Multidisciplinary Approach", HP Technologies,2003.

Webliography:

1. https://www.geeksforgeeks.org/software-testing-tutorial/
2. https://www.guru99.com/software-testing.html
3. https://www.guru99.com/junit-tutorial.html

SWAYAM/NPTEL/MOOCs Certification:

1. Course Name : Software Testing By Prof. Meenakshi D'souza, IIIT Bangalore
   https://onlinecourses.nptel.ac.in/noc22_cs61/preview
2. Course Name : Foundations of Software Testing and Validation
   https://www.coursera.org/learn/foundations-of-software-testing-and-validation
3. Course Name : Software Testing and Automation Specialization
   https://www.coursera.org/specializations/software-testing-automation

PRACTICE PROBLEMS

S.No	Problem
1.	A program reads three integer values, representing the lengths of the sides if the triangle. The program prints whether the triangle is scalene, isosceles or equilateral. Develop a set of test cases that would test the program adequately.
2.	Derive a flow graph for the above program and apply basis path testing to develop test cases that will guarantee the execution of all the statements. Execute the cases and show the results.
3.	Write a program in any programming language, to accept 10 numbers & sort them in the order accepted at run time and design test cases for the condition testing. Also mention the expected results.
4.	Write programs for binary and linear search and define the test cases for each program to find numbers in an array.
5.	Write a program to find the sum of the matrices. Write all the test cases so as to verify the correctness of the logic.
6.	Write a program to create fibonacci series and write the test cases for it.

7.	Create du and dc graph for the following program: scanf(x,y);
	if (y < 0)
	pow = pow – y;
	else
	pow = y;
	
	z = 1.0;
	while(pow != 0)
	{
	z = z * x;
	pow = pow – 1;
	}
	if (y < 0)
	z = 1.0/z; printf(z);

8.	Define DU path for the following function:
	public int gcd(int x, int y){
	int tmp;
	while(y!=0){

	tmp = x % y;
	x = y;
	y = tmp;
	}
	return x;
	}

9.	Write a program to compute the factorial of a number and create du and dc graph for the same.
10.	Create the graph matrix and compute the cyclomatic complexity for a program to find factorial of a number.
11.	Write a program to create fibonacci series and create du and dc graph for the same.

12.	Find the Cyclomatic Complexity for the below mentioned code:
	i = 0;
	n=4; //N-Number of nodes present in the graph
	while (i<n-1) do
	j = i + 1;
	while (j<n) do
	if A[i]<A[j] then
	swap(A[i], A[j]);
	end do;
	j=j+1;
	end do;

13.	Calculate cyclomatic complexity for the given code: IF A = 354 THEN IF B > C THEN A = B ELSE A = C END IF END IF PRINT A

14.	Calculate cyclomatic complexity for the given code:
	{ int i, j, k;
	for (i=0 ; i<=N ; i++)
	p[i] = 1;
	for (i=2 ; i<=N ; i++)
	{
	k = p[i]; j=1;
	while (a[p[j-1]] > a[k] {
	p[j] = p[j-1];
	j--;
	}
	p[j]=k;
	}

15.	Prepare a comprehensive checklist to test a WEB Site

16.	A university's web site allows students to enroll online bio-data. The form contains following fields: i. Name of the student ii. Father's name iii. Address iv. City v. State vi. Pin code vii. Sex viii. Date of Birth ix. Academic Qualifications a. Exam Passed b. University/Board c. Marks obtained d. Division e. Max Marks Design the validation checks for the given fields
17.	Consider an application that accepts a numeric number as input with a value between 10 to 100 and finds its square. Now, using equivalence class testing, what can be its equivalence classes?
18.	Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of boundary value, derive test cases, execute these test cases and discuss the test results. Assumption price for lock=45.0, stock=30.0 and barrels=25.0 production limit could sell in a month 70 locks,80 stocks and 90 barrels commission on sales = 10 % <= 1000 and 15 % on 1000 to 1800 and 20 % on above 1800.
19.	Design, develop, code and run the program in any suitable language to implement the binary search algorithm. Determine the basis paths and using them derive different test cases execute these test cases and discuss the test results.
20.	Design, develop, code and run the program in any suitable language to implement the quicksort algorithm. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.
21.	Design, develop, code and run the program in any suitable language to implement an absolute letter grading procedure, making suitable assumptions. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results
22.	Design, develop, code and run the program in any suitable language to implement the NextDate function. Analyze it from the perspective boundary value testing and equivalence class analysis. Derive different test cases, execute these test cases and discuss the test results.
23.	Write a Java Program to implement the methods next() and hasNext() in the class StringToWords (found in ps1.warmup). Its constructor takes a String as argument, and next() should return the sequence of space-separated words in the string. Assume that behavior is undefined for more than one space characters between words, and non-space whitespace characters (like tab or newline) are not considered spaces. Eclipse comes with a debugger which allows you to execute your code one statement at a time, see the values of variables, and stop the execution of the program at any point.

24.	Create Junit program to create test cases to check a test string for “Anagram”, “Isogram” and “Panagram”.
25.	Write a Java program using Junit that verifies if the string variable and string passed in the condition are both equal.
26.	Write a JUnit test assuming you have two StringBuffer references named sbOne and sbTwo and you only want it to pass if the two refernces point to the same StringBuffer object.
27.	Write a JUnit test assuming you have an array of int valus and you only want the JUnit test to fail if any of the values are less than 20.
28.	Write a Junit program that returns true in case the triangle is a valid triangle, i.e. with the provided sides a, b and c a triangle can be constructed. Two sides combined need to be smaller or equal to the third, and each side needs to be longer than 0.
29.	Write a script to open google.co.in using chrome browser (ChromeDriver).
30.	Write java script in selenium to login into a website.
31.	Write and test a program to provide total number of objects present/ available on the page.
32.	Write and test a program to get the number of list items in a list / combo box.
33.	Write and test a program to count number of check boxes on the page checked and unchecked count.
34.	Write a Script in Selenium to ensure the following conditions in a Web Form: a. Name field can’t be blank b. Subject field must be between 5 and 100 characters c. Form can’t be submitted with invalid entries.