Index

Name : M. Sukish Roll No 221801053

Subject : .... Std : .... Sec : ....

School : ....

S.No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	24/7/24	Development of Requirement specification		[illegible]
2.	24/7/24	Function oriented design using SA/SD		[illegible]
3.	07/8/24	SRS - E commerce Product catalogue Analysis		[illegible]
4	20/8/24	object oriented Design using UML		[illegible]
5	21/8/24	Test case Designing		[illegible]
6.	24/8/24	The implementation using JAVA		[illegible]
7.	24/8/24	Testing		[illegible]
8	6/10/24	configuration management tools		[illegible]
9	6/10/24	Program Analysis Tools		[illegible]

Completed [illegible]

Palaniyappa

• Add actions, control flows, decisions and merge nodes

6 State Machine diagram

• create a state machine diagram to represent the states and transitions of particular class or component

• Add states, transitions, events and Actions

Result :
Thus we have learnt the clear and concise software requirement and specificati

Ex No: 2
24/7/24

FUNCTION ORIENTED ~~DE~~ ~~TS~~ SPECIFICATION USING SA/SD

log

Aim:
To supply structured analysis and design techniques to software design.

Procedure:
step by step Instructions:

1. Introduction to SA/SD:
   - understand the concepts of structured Analysis (SA) and structured Design (SD)
2. create Data Flow Diagrams (DFD):
   - Identify and document the major processes of the system.
   - create context-level DFD and level 1 DFD
3. Design using structured charts
   - Break down the processes into smaller modules.
   - design the structured charts showing the hierarchy & relationships.

[illegible]

Result:
Therefore the structured Analysis and design techniques to software design was to applied and created.

Conclusion:

Thus, we have designed the software Requirement specification for product catalog Analysis successfully documented.

[illegible]

Step 4: create class diagram.

1. Identify classes and Relations

* classes: user, Product, cart, order.

* Relationships: user has cart, cart contains Product, order is created from cart

2. Draw class Diagram:

* use UML tools to create a class diagram showing the structure of the system.

Step 5: create sequence Diagram:

1. Identify Interactions:

* Interactions for use cases like register, login and checkout.

2. Draw the sequence diagram:

* use UML tools to create sequence diagrams for the main interactions.

Result: Thus object oriented Design principles using UML are applied and implemented

Example code and output
Test Case Execution in Java
Test case Runner.java

```
Public class Test case Runner {
  Public static void main(String[] args){
    System.out.Println("Executing Test
case 1: User Registration with valid data");
boolean result1 = testUserRegistration
("valid User", "valid@example.com",
"valid Password");
System.out.Println("Test case1 Result:
"+(result1?"Passed"));
System.out.Println("Executing Test
case2: User Registration with Invalid
Email");
System.out.Println("Test case 2 Result:"+
(result2?"Passed":"Failed"));
}
Public static boolean testUserRegistration(String
username, String email, String Password)
{
if(!email.contains("@")){
  System.out.Println("Invalid email
  format");
return false; }
System.out.Println("User registered
successfully");
return true; }}
```

[illegible]

Result:
Therefore the effective test cases for software testing was learned and designed.

```
Public void removeProduct(Product
Product) {
  Product.remove(product);
 }
Public double getTotalPrice() {
  double total=0;
 for( Product product: products) {
  }
 return total;
 }
Public static void main(String[] args) {
Product product1=new Product("laptop",999.99)
Product product1=new Product("Phone",499.99)
ProductProduct2=newProduct
Shopping cart cart=newShoppingCart();
cart.addProduct(Product1);
cart.addProduct(Product2);
system.out.println("Total Price: $"+cart.get
TotalPrice());
cart.remove(product2);
system.out.println("Total Price after removing
Phone: $"+cart getTotalPrice()); }}
```

5. compile & Run The Application:
java Product.java ShoppingCart.java
java shopping Cart

[illegible]

Result: The simple java application That demonstrates basic object-oriented programming concepts such as creating classes, instantiating objects and invoking methods was implemented.

```
@Test(expected=IllegalArgumentException.class)
Public void testDivideByZero() {
    Calculator calc = new Calculator();
    calc.divide(6,0);
  }
}
```

Running Tests and Results

1. Run Tests
- use The test runner in your IDE or run mvn Test if using Maven.

2. Excepted Results.
- All test cases should pass.
- you should see output indicating The successfull execution of tests e.g.

[illegible]

Results: Therefore, the comprehensive test cases using automated testing tools to ensure ~~the~~ quality and reliability of software applications was developed and executed.

5. Push to Remote Repository

```
git remote add origin <your-repo-url>
git push -u origin main
```

Expected Results

1. UML Diagrams in StarUML:

2. Version control With Git:

- A fully initialized Git repository tracking changes to your UML diagrams, including class, use case, and to your UML diagrams.
- Properly managed branches for different features.
- Successfully merged changes into the main branch.
- A remote repository containing your project files.

[illegible]

Results:

Therefore, the CASE tools and configuration management tools for software modeling and version control in a structured software deloPment process was effectively us

Step5: Run Sonar Qube Analysis

Step6: Analyze The Results

Expected Results:

1. Sample Class.java:

2. Sonar Qube Analysis:

Sonar Qube Dashboard:

- Bugs
- code Smells:
- vulnerabilities

By following this procedures you will utilize program Analysis tools to improve code quality and main maintainability in your software development process.

Result:
Therefore, The program Analysis tools for identifying and addressing code quality issues in software development was utilized.