Index

[1.](#_heading=h.30j0zll) Introduction 2

[1.1 Sprint Objectives 2](#_heading=h.1fob9te)

[1.2 Outcome 2](#_heading=h.3znysh7)

[2.](#_heading=h.2et92p0) Usecase selection guidelines 2

[3.](#_heading=h.tyjcwt) Execution 2

[2.1 Execution Steps 2](#_heading=h.3dy6vkm)

[2.2 Template Docs 2](#_heading=h.1t3h5sf)

[2.3 Sprint Connect 2](#_heading=h.4d34og8)

[3. Assessment Parameters 2](#_heading=h.2s8eyo1)

[Appendix 2](#_heading=h.17dp8vu)

[Annexure A Sample Review Comments 2](#_heading=h.3rdcrjn)

[Annexure B Sample Directory structure and Makefile 2](#_heading=h.26in1rg)

# Introduction

## 1.1 Sprint Objectives

* Opportunity to apply your learnings to solve a real-world problem and demonstrate your acquired technical and soft skills.
* Understand the Software development process, design, develop and deliver a maintainable software in a structured way following process documents and guidelines.

## 1.2 Outcome

* Pax should be able to understand given requirements, design, develop and deliver a bug free software (with 0 defects, warnings, and memory leak) using development tools in a structured way following process.
* Should be aware of process documents to use and guidelines to follow while developing software.
* Should be able to understand/update process documents, use the development tools (linux vi editor, gcc, makefile, valgrind, static analyzer, git, gcov etc).

# Usecase selection guidelines

| **Variant** | **Sprint** | **Sprint UseCase** |
| --- | --- | --- |
| Systems C with Linux System programming | Sprint 1 & 2 | **A multithreaded client server application software** and **C using data structures**. Some examples are: |
| Telecom domain (protocol specific say dns, ftp, telnet etc)  Chat server (to use select() system call) |
| Line editor |
| Tools (network monitoring, performance monitoring, dynamic memory management, etc) |
| File parser (hosts, routing table, data logger etc) |
| System logger |
| File search engine (Local, Remote) |
|  |
| Systems CPP with Linux System programming | Sprint 1 & 2 | A multithreaded client server application software in **CPP using design patterns and STL**. Some examples are: |
| Telecom domain (protocol specific say dns, ftp, telnet etc)  Chat server (to use select() system call) |
| Line editor |
| Tools (network monitoring, performance monitoring, dynamic memory management, etc) |
| File parser (hosts, routing table, data logger etc) |
| System logger |
| File search engine (Local, Remote) |
|  |
| Systems C CPP Programming on Linux | Sprint 1 | An application software involving **multithreading and data structure in C**. Some examples are: |
| Line editor |
| Tools (network monitoring, performance monitoring, dynamic memory management, etc) |
| File parser (hosts, routing table, data logger etc) |
| System logger |
| File search engine (Local) |
|  |
|  | Sprint 2 | An application software involving **multithreading and data structure in CPP**.  Some examples are: |
| Line editor |
| Tools (network monitoring, performance monitoring, dynamic memory management, etc) |
| File parser (hosts, routing table, data logger etc) |
| System logger |
| File search engine (Local) |
| Code migration from C++ to C++11 |
|  |

# Execution

## 2.1 Execution Steps

Create a group in MS teams with name as below and with below substructure as

in table.

*<StartDate>\_<Batch Name>\_SPRINT<Y>\_GROUP<XX>*

*where*

*Y- 1 or 2*

*XX- Group ID as 01/02 ..*

| Plan | Sprint Execution Plan   | Activity | Expected Start Date | Expected End Date | Actual Start Date | Actual End Date | | --- | --- | --- | --- | --- | | SRS- Release |  |  |  |  | | Design |  |  |  |  | | Coding |  |  |  |  | | UT Plan |  |  |  |  | | UT |  |  |  |  | | IT Plan |  |  |  |  | | IT |  |  |  |  | | Sprint Demo |  |  |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SRS | SRS – Draft and Release documents |
| Storyboard | Refer <http://productleaders.org/posts/how-to-create-a-story-board/>  And prepare a story UI |
| Design | Word Documents   | Design Draft  Design Release | Sec 1,2, 3  Introduction - Purpose, Scope, Audience, Functional overview  Design Overview - High Level diagram (Level 0 DFD), alternate design strategy  System Architecture – High level architecture diagram, Low level module interaction diagram, other Language specific artefacts as below.  C based sprint: Level 1 DFD etc, global data structures, functions, pseudocode for complex functions  Refer [Level X Diagrams](https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/)  C++ based sprint: class diagram, Usecase Diagram, ER Diagram, activity, sequence diagram, STL containers and algorithms, pseudocode for complex functions  Hint for Design: High level Architecture Diagram showing all modules and their interaction🡪Submodule identification🡪submodule functions /class with members, data structures and algorithms /STL container + STL algorithms + database if required  For C sprints may use third party libraries like [glib](https://docs.gtk.org/glib/index.html) for data structures (List, Queue, Hash Table etc) | | --- | --- | | Design Review Checklist |  | | Design Review Log | Capture the peer-review comments of Design categorising the defects | |
| CUT | Subfolders Description   | Code | Source code | | --- | --- | | Code Inspection Log | Code review checklist  Code review log | | Tools Report | [gcov](https://docs.oracle.com/en/operating-systems/oracle-linux/6/porting/ch02s05s01.html)\_report  Splint/CPPCheck or any static analyzer report  Valgrind report  Code to be checked in git | | UT\_IT\_Plan\_Reports | UT and IT Plan and Reports containing UT/IT Test cases (Sunny and Rainy test cases added and defects if any identified in each test cycle)  Refer [FAQ on Testing](http://sqa.fyicenter.com/FAQ/Testing-Techniques/What_is_the_BEST_WAY_to_write_test_cases_.html)  [positive/Sunny-and-negative/Rainy-test-scenarios](https://www.softwaretestinghelp.com/positive-and-negative-test-scenarios/) | |
| RTM | Requirements Traceability Matrix - a document that shows the mapping between requirements and other artifacts.   | Req | Design Mapping | Code Mapping | UT Mapping | IT Mapping | | --- | --- | --- | --- | --- | | REQ1.1 | 1.1 | File/function reference | UT Testcase ID | IT Test Case ID | |
| MOM | A single running document capturing each MOM |

## 2.2 Template Docs

* Minutes of Meeting
* QMS\_UT\_IT TestPlan\_Reports
* Inspection/Review Log
* Review Checklist for Design and Code
* RTM
* Schedule plan

## 2.3 Sprint Connect

| **Sprint Connects with LoT Lead (Mail/Meeting)** | **When (Considering D1- start of Sprint)** | **Expected Details** |
| --- | --- | --- |
| Connect 1 – mail | 3 working days before D1 of sprint | Sprint Use case Documents(Word/pdf/excel) with clearly defined requirements |
| Connect 2- mail | After Design (Should be within D1+3) | All phase specific artefacts  + updated sprint tracker |
| Connect 3 – mail | After Coding (Should be within D1+6) | All phase specific artefacts + updated sprint tracker |
| Connect 4 – Mock Demo Meeting | After UT/IT (atleast 2 days before final assessment ) | All phase specific artefacts + updated sprint tracker + presentation |

# 3. Assessment Parameters

**Personal Skills**

* + Presentation skills
  + Communication/Interpersonal Skills
  + Alertness
  + Grasping power etc.,

**Technical Skills**

* + SRS understanding
  + Availability of phase wise artefacts
  + Follow of process (use of checklist & review for Design and Code, Coding guidelines for Code, use of tool for coding & unit testing, generation of tool reports)
  + Code quality (Followed guidelines, Comments added, modular, multifile, exceptions handled, optimized etc.,)
  + Ability to handle complexity
  + Completeness w.r.t to timelines as per requirements
  + Working Demo

Appendix

Annexure A Sample Review Comments

**General**

* *C++ functions used are very less, more C library functions in use*
* *Exception are handled in C syntax instead of try..catch*
* *Use STL containers instead of linked list*
* *Handle multiple client connections using select*

[*https://www.ibm.com/docs/en/zos/2.4.0?topic=calls-select*](https://www.ibm.com/docs/en/zos/2.4.0?topic=calls-select)

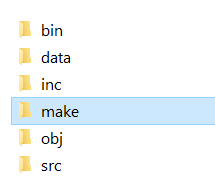
* *Application should not continue after error*
* *missing directory structure and makefile*
* *have used statistical methods - Good*
* *Good design*
* *Menu options could have been better*
* *No hardcodings for switch case, array dimension etc., rather use macro*
* *malloc error handling missing*
* *scope for code optimization*
* *Failure exit handling - improve*
* *Very few UT/IT test cases*
* *Be aware of sunny and rainy test cases*
* *.h file not as per guideline (use #ifndef…. to avoid multiple inclusion)*
* *do not mix C and CPP library calls like new and malloc*
* *For CPP sprint use STL containers rather than linked list*
* *Usecase diagram to be updated*
* *flow chart to be updated*
* *Use proper error messages*
* *function return to be checked*
* *use macros for fixed and read only strings like file name, path etc*
* *do cleanup on error exit*
* *test client without server*
* *Remove unwanted prints*
* *improve the modularity*
* *Good unit testcases using CUnit*
* *Good makefile and good demo*
* *Scope for code optimization*
* *Use valid port numbers in the in the range 49152 to 65535*
* *do not display password*
* *include SIGSEGV signal also*
* *Add block comments*
* *mutex lock should be for minimal duration*
* *File open errors to be handled*
* *Close all opened files*
* *Know the difference between CU\_ASSERT and CU\_FAIL*
* *Readability could have been improved*
* *Update Level 0 context diagram*
* *Global data structures to be identified*
* *Functions to have valid returns and caller to check return status and proceed*
* *better file names, function name, class name*
* *function size is too big*
* *Scope for code optimization*
* *Use perror and handle system errors*
* *Use \_\_FUNCTION\_\_ instead of \_\_func\_\_*
* *Use #ifndef to avoid multiple .h includes*

***Project Specific***

* *use case diagram should not be with mutliple users on both side*
* *C++ functions used are very less , more C library functions in use*
* *Exception are handled in C syntax instead of try..catch*
* *Use STL containers instead of linked list*
* *Use Consistent names‌ menu , menu item etc*
* *Check the error message displayed for a new user-it is incorrect*
* *Application should not continue after error*
* *Items with qty 0 should not be displayed---test it thoroughly*
* *Each item to have a unique itemId*
* *Issues with item add and cancel*

Annexure B Sample Directory structure and Makefile

Project Directory should contain below sub directories



*Makefile to be in make directory.*

***Sample makefile for .c files:***

*INC = ../inc*

*SRC = ../src*

*BIN = ../bin*

*OBJ = ../obj*

*CFLAGS = -c -g -Wall*

*LFLAGS = -o*

*GCC = gcc*

*IFLAGS = -I $(INC)*

*CVFLAGS = -v --tool=memcheck --leak-check=full --show-reachable=yes --log-file=valclient*

*all : $(BIN)/client $(BIN)/server*

*$(OBJ)/clientprg.o : $(SRC)/clientprg.c*

*$(GCC) $(CFLAGS) $(SRC)/clientprg.c $(IFLAGS)*

*mv \*.o ../obj*

*$(OBJ)/clientutility.o : $(SRC)/clientutility.c*

*$(GCC) $(CFLAGS) $(SRC)/clientutility.c $(IFLAGS)*

*mv \*.o ../obj*

*$(BIN)/clientprg : $(OBJ)/clientprg.o $(OBJ)/clientutility.o*

*$(GCC) $(OBJ)/clientprg.o $(OBJ)/clientutility.o $(LFLAGS) $(BIN)/clientprg*

*clean:*

*rm -f $(OBJ)/\*.o $(BIN)/clientprg*

*valgrindclient:*

*valgrind $(CVFLAGS) $(BIN)/clientprg 10.203.161.8 57319 anmol*

*mv valclient $(BIN)/*

***Sample makefile for .cpp files:***

*INC = ../inc*

*SRC = ../src*

*BIN = ../bin*

*OBJ = ../obj*

*CFLAGS = -c -g -Wall*

*LFLAGS = -o*

*GCC = g++*

*IFLAGS = -I $(INC)*

*CVFLAGS = -v --tool=memcheck --leak-check=full --show-reachable=yes --log-file=valclient*

*all : $(BIN)/client $(BIN)/server*

*$(OBJ)/clientprg.o : $(SRC)/clientprg.cpp*

*$(GCC) $(CFLAGS) $(SRC)/clientprg.cpp $(IFLAGS)*

*mv \*.o ../obj*

*$(OBJ)/clientutility.o : $(SRC)/clientutility.cpp*

*$(GCC) $(CFLAGS) $(SRC)/clientutility.cpp $(IFLAGS)*

*mv \*.o ../obj*

*$(BIN)/clientprg : $(OBJ)/clientprg.o $(OBJ)/clientutility.o*

*$(GCC) $(OBJ)/clientprg.o $(OBJ)/clientutility.o $(LFLAGS) $(BIN)/clientprg*

*clean:*

*rm -f $(OBJ)/\*.o $(BIN)/clientprg*

*valgrindclient:*

*valgrind $(CVFLAGS) $(BIN)/clientprg 10.203.161.8 57319 anmol*

*mv valclient $(BIN)/*

About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of 270,000 team members in nearly 50 countries. With its strong 50 year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2020 global revenues of €16 billion.

Get the Future You Want | [www.capgemini.com](http://www.capgemini.com)

