

Teaching Guidelines for
Logic Building & Problem Solving (optional)
PG-DAC September 2022

Duration: 18 hours online sessions + problem solving as self study

Objective: This preparatory module is conducted online before the actual PG-DAC course commences for students to learn to think logically and how to solve problems. A number of problems will be given to the students to solve them logically.

Prerequisites: Knowledge of computer fundamentals.

Evaluation: No separate evaluations. Daily assignment problems are to be solved and submitted.

Note: Each day comprises 3 hours' online lecture sessions followed by solving various problems using the topics learnt by the students.

Day 1: Building Logic and Solving Problems

Lecture:

Logical thinking
Problem solving process

Assignments:

Practice logic building on given problems

Day 2: Decision making

Lecture:

If-else, nested if-else, switch case

Assignments:

Decision making problems using if-else,
nested if-else, switch case

Day 3: Control Statements

Lecture:

Loops

Assignments:

Conditional problems using loops

Day 4: Modular Programming

Lecture:

Functions

Assignments:

Problems using functions

Day 5: Arrays

Lecture:

Arrays

Assignments:

Problems using arrays

Day 6: Strings

Lecture:

Strings

Assignments:

Problems using strings

Teaching Guidelines for Concepts of Programming & Operating System PG-DAC September 2022

Duration: 72 hours (38 theory hours + 26 lab hours + 8 revision/practice hours)

Prerequisites: Knowledge of computer fundamentals

Evaluation: 100 marks (Concepts of Programming – 40 marks + Operating Systems – 60 marks)

Weightage: Theory exam – 40%, Lab exam – 40%, Internals – 20%

Concepts of Programming

Duration: 24 hours (12 theory hours + 12 lab hours)

Objective: To introduce the fundamental programming concepts in Java.

Evaluation: 40 marks (Theory exam: 12 + Lab exam: 20 + Internals: 8 marks)

Text Book:

- Core and Advanced Java Black Book / Dreamtech Press

References:

- Java The Complete Reference by Herbert Schildt / McGraw Hill
 - Core Java : Fundamentals - Volume 1 Gary Cornell, Cay S. Horstmann/ Pearson
 - Programming in Java by Sachin Malhotra, Saurabh Choudhary / Oxford University Press
-

(Note: Each Session is of 2 hours)

Sessions 1 & 2:

Lecture:

Getting Started

- Setup development environment (JRE, JDK, eclipse)
- Writing your first Java program

Variables & Methods

- About main () method
- Java Data Types, Primitives and Binary Literals
- Data type compatibility and casting of primitive data types
- Static variables and methods
- Accessing static variables and methods of different class
- Final variables

Operators

- Arithmetic Operator
- Relational Operator
- Logical Operator
- Unary Operator
- Ternary Operator
- Assignment Operator

Lab:

Write Java programs to:

- Print Hello World
- Add two numbers/binary numbers/characters
- Calculate compound interest
- Calculate power of a number
- Swap two numbers
- Calculate area of rectangle
- Calculate area and circumference of circle using multiple classes
- Java program to find ASCII value of a character
- Print default values of primitive data type variables
- Swap two variables without using the third variable
- Print Fibonacci series till n

Session 3: Conditional and Looping Statements

Lecture:

- If, else if, switch
- break & continue keyword
- for loop
- while loop
- do while loop
- Recursion

Lab:

Write Java programs to:

- Display prime numbers between 1 and 100 or 1 and n
- Find the factorial of a number
- Check if a number is palindrome or not
- Add two integer variables in 5 different ways using functions and control statement
- Find square root of a number without sqrt method
- Check Armstrong number
- Calculate grades of students using their marks
- Use switch case, recursion, print patterns, etc.

Session 4: Objects

Lecture:

- Reference variables and methods
- Constructors (Default constructor, parameterised constructor)
- Static method v/s instance method
- Reference variable as instance member of the class
- String class

Lab:

- Build a class Employee which contains details about the employee and compile and run its instance.
- Build a class which has references to other classes. Instantiate these reference variables and invoke instance methods.

Session 5 & 6: Arrays

Lecture:

- Initializing an Array in Java
- Two dimensional array in java
- Java Variable Arguments explained
- Add, update, read array elements
- Sorting and searching in array

- Java String Array to String
- How to copy arrays in Java

Lab:

Write Java programs to:

- Calculate average of numbers using Array
- Reverse an array
- Sort an array in ascending order
- Convert char Array to String
- Add two Matrix using Multi-dimensional Arrays
- Sort strings in alphabetical order
- Find out the highest and second highest numbers in an array
- Concatenate two arrays

Concepts of Operating System

Duration: 40 hours (26 theory hours + 14 lab hours)

Objective: To introduce Operating System concepts with Linux environment, and to learn Shell Programming.

Evaluation: 60 marks (Theory exam: 28 + Lab exam: 20 + Internals: 12 marks)

Text Books:

- Operating Systems Principles by Abraham Silberschatz, Peter Galvin & Greg Gagne / Wiley
- Unix Concepts and Applications by Sumitabha Das / McGraw Hill

References:

- Modern operating Systems by Andrew Tanenbaum & Herbert Bos/ Pearson
- Principles of Operating Systems by Naresh Chauhan / Oxford University Press
- Beginning Linux Programming by Neil Matthew & Richard Stones / Wrox
- Operating System : A Design-Oriented Approach by Charles Crowley / McGraw Hill

(Note: Each Session is of 2 hours)

Session 1:**Lecture:***Introduction to OS*

- What is OS; How is it different from other application software; Why is it hardware dependent
- Different components of OS
- Basic computer organization required for OS
- Examples of well known OS including mobile OS, embedded system OS, Real Time OS, desktop OS server machine OS etc. ; How are these different from each other and why
- Functions of OS
- User and Kernel space and mode; Interrupts and system calls

(No Lab)**Session 2:****Lecture:***Introduction to Linux*

- Working basics of file system

- Commands associated with files/directories & other basic commands. Operators like redirection, pipe
- What are file permissions and how to set them
- Permissions (chmod, chown, etc); access control list; network commands (telnet, ftp, ssh, sftp, finger)
- System variables like – PS1, PS2 etc. How to set them

Shell Programming

- What is shell; What are different shells in Linux?
- Shell variables; Wildcard symbols
- Shell meta characters; Command line arguments; Read, Echo

Lab:

- Working with various OS commands
- Shell programs related to Session 2

Session 3:

Lecture:

Shell Programming

- Decision loops (if else, test, nested if else, case controls, while...until, for)
- Regular expressions; Arithmetic expressions
- More examples in Shell Programming

Lab:

- Shell Programs related to Session 3

Sessions 4, 5 & 6:

Lecture:

Processes

- What is process; preemptive and non-preemptive processes
- Process management; Process life cycle
- What are schedulers – Short term, Medium term and Long term.
- Process scheduling algorithms – FCFS, Shortest Job First, Priority, RR, Queue. Belady's Anomaly
- Examples associated with scheduling algorithms to find turnaround time to find the better performing scheduler.
- Process creation using fork; waitpid and exec system calls; Examples on process creation; Parent and child processes
- Orphan and zombie processes

Lab: (4 hours)

- Creating processes - parent and child processes
- Handling orphan and zombie processes.

Session 7:

Lecture:

Signals

- What are signals
- Generating and handling signals

Threads

- What are threads; user and kernel threads; how threads are different from processes
- Thread programming using pthread.

Lab:

- Assignment on signals
- Assignment on threads – Thread creation, thread synchronization

Sessions 8 & 9:

Lecture:

Memory management

- What are different types of memories; What is the need of Memory management
- Continuous and Dynamic allocation
- First Fit, Best Fit, worst Fit
- Compaction
- Internal and external fragmentation
- Segmentation – What is segmentation; Hardware requirement for segmentation; segmentation table and its interpretation
- Paging – What is paging; hardware required for paging; paging table; Translation look aside buffer
- Concept of dirty bit
- Shared pages and reentrant code
- Throttling

(*No Lab*)

Session 10:

Lecture:

Virtual Memory

- What is virtual memory
- Demand paging
- Page faults
- Page replacement algorithms

(*No Lab*)

Session 11:

Lecture:

Deadlock

- Necessary conditions of deadlock
- Deadlock prevention and avoidance
- Semaphore
- Mutex
- Producer consumer problem
- Dead-lock vs Starvation

Lab:

- Semaphore, Mutex

Sessions 12 & 13:

Lecture:

Inter process communication

- Message queues,
- Shared memory
- Pipes
- FIFO

Lab: (2 hours)

- IPC using shared memory
- IPC using Pipes
- IPC using FIFO

Teaching Guidelines for Object Oriented Programming with Java

PG-DAC September 2022

Duration: 104 hours (48 theory hours + 48 lab hours + 8 revision/practice hours)

Objective: To reinforce knowledge of Object Oriented Programming concepts using Core Java.

Prerequisites: Basic knowledge of computer programming

Evaluation: Total 100 marks

Weightage: Theory exam – 40%, Lab exam – 40%, Internals – 20%

Text Book:

- Core and Advanced Java Black Book / Dreamtech Press

References:

- Java 8 Programming Black Book / Dreamtech Press
- Core Java : Volume 1 - Fundamentals by Cay S. Horstmann / Prentice Hall
- Core Java : Volume 2 - Advanced Features by Cay S. Horstmann / Prentice Hall
- Programming in Java by Sachin Malhotra, Saurabh Choudhary / Oxford University Press
- Java The Complete Reference by Herbert Schildt / McGraw Hill
- Core Java 8 for Beginners by Sharanam Shah, Vaishali Shah / Shroff Publishers
- Murach's Java Programming by Joel Murach / Mike Murach
- Object-Oriented Analysis and Design with applications by Grady Booch / Pearson
- Object-Oriented Analysis and Design Using UML - An Introduction to Unified Process and Design Patterns by Mahesh P. Matha / PHI

(Note: Each Session is of 2 hours)

Session 1:

Lecture:

Introduction to java

Features of java

JVM Architecture

JDK and its usage

Structure of java class

Working with data types: Primitive data types

Session 2:

Lecture:

Operators

- Unary, binary, Arithmetic, Assignment, compound, relational, logical, equality

Control statements (*optional*)

- if-else-if, switch, ternary operator, for loop, while loop, do-while loop

Declaring variables and methods

Data type compatibility

Lab 1 & 2:

Get yourself acquainted with java environment.

Print different patterns of asterisk (*) using loops (e.g. triangle of *).

Tutorial:

Compare syntactical similarities and dissimilarities between Java and C++.

Session 3:**Lecture:**

Static variables and methods

Accessing static variables and methods of different class

Introduction to reference data types

Reference variables and methods

Difference between reference data types and primitive data types

Difference between reference variable and static variable

Session 4:**Lecture:**

Constructors, initializing reference variables using constructors

Pass by value v/s pass by reference

Re-assigning a reference variable

Passing reference variable to method

Initializing reference variable of different class

Heap memory and stack memory

Lab 3 & 4:

Print default values of static & instance variables for different data types.

Build a class Employee which contains details about the employee and compile and run its instance.

Build a class which has references to other classes. Instantiate these reference variables and invoke instance methods.

Tutorial:

Understand role of stack and heap memory in method invocation and object creation.

Object Oriented Programming Concepts**Session 5:****Lecture:**

Introduction to OOP concepts

Encapsulation

Inheritance: single & multilevel

Session 6:**Lecture:**

Inheritance: Hierarchical

Association, Aggregation and Composition

Polymorphism: Compile time and runtime polymorphism

Rules of overriding and overloading of methods

super and this keywords

Lab 5 & 6:

Create a class Employee and encapsulate the data members.

Create demo applications to illustrate different types of inheritance.

Session 7:**Lecture:**

Upcasting & downcasting of a reference variable

Abstract class and abstract methods

Interface (implementing multiple interfaces)

Session 8:**Lecture:**

Final variables, final methods and final class

Functional interface

New interface features (Java 8 & above)

Arrays

Enumerations

Lab 7 & 8:

Create an Array of Employee class and initialize array elements with different employee objects.

Try to understand the no of objects on heap memory when any array is created.

Session 9:**Lecture:**

Access modifiers (public, private, protected and default)

Packages and import statements

Static imports

Constructor chaining (with and without packages)

Accessing protected variables and methods outside the package

Session 10:**Lecture:**

Garbage collection in java

Requesting JVM to run garbage collection

Different ways to make object eligible for garbage collection: (Nulling a reference variable, Re-assigning a reference variable & island of isolation)

Finalize method

Lab 9 & 10:

Create a demo application to understand the role of access modifiers.

Implement multilevel inheritance using different packages.

Access/invoke protected members/methods of a class outside the package.

Override finalize method to understand the behavior of JVM garbage collector.

Sessions 11 & 12:**Wrapper Classes and String Class****Lecture:**

Wrapper classes and constant pools

String class, StringBuffer & StringBuilder class

String pool

Lab 11 & 12:

Create sample classes to understand boxing & unboxing.

Use different methods of java defined wrapper classes.

Create StringDemo class and perform different string manipulation methods.

Tutorial:

Understand the difference between String / StringBuffer / StringBuilder.

Sessions 13 & 14:

Exception Handling

Lecture:

Exception hierarchy, Errors, Checked and un-checked exceptions

Exception propagation

try-catch-finally block , throws clause and throw keyword

Multi catch block

Creating user defined checked and unchecked exceptions

Lab 13 & 14:

Create user defined checked and unchecked exceptions .

Session 15:

java.io & java.nio Package

Lecture:

Brief introduction to InputStream, OutputStream, Reader and Writer interfaces

NIO package

Serialization and de-serialization

Shallow copy and deep copy

Lab 15:

Create a Demo class to Read & write image/text files.

Create SerializationDemo class to illustrate serialization and de-serialization process.

Session 16:

Lecture:

Object Class & java.util Package

Date, DateTime, Calendar class

Converting Date to String and String to Date using SimpleDateFormat class

Object Class: Overriding to String, equals & hashCode method

Collections

Session 17 & 18:

Lecture:

Introduction to collections: Collection hierarchy

List, Queue, Set and Map Collections

List Collection:

- ArrayList, LinkedList
- Vector (insert, delete, search, sort, iterate, replace operations)

Collections class

Comparable and Comparator interfaces

Queue collection

Lab 16, 17 & 18:

Create DateManipulator class to convert String to date, date to String and to find out number of days between two dates.

Create a list of java defined wrapper classes and perform insert/delete/search/iterate/sort operations.

Create a collection of Employee class and sort objects using comparable and comparator interfaces.

Implement Queue data structure using LinkedList and Queue collection.

Sessions 19 & 20:

Lecture:

Set Collection:

- HashSet, LinkedHashSet & TreeSet collection
- Backed set collections

Map Collection:

- HashTable, HashMap, LinkedHashMap & TreeMap classes
- Backed Map collections

Generics

Concurrent collections

Lab 19 & 20:

Create an Employee HashSet collection and override equals & hashCode methods to understand how the set maintains uniqueness using these methods.

Create a Sample class to understand generic assignments using "? extends SomeClass" , "? super someclass " and "?".

Session 21:

Lecture:

MultiThreading : Thread class and Runnable Interface

sleep, join, yield, setPriority, getPriority methods

ThreadGroup class

Lab 21:

Create multiple threads using Thread class and Runnable interfaces.

Assign same task and different task to multiple threads.

Understand sleep, join, yield methods.

Sessions 22 & 23:

Lecture:

Synchronization

Deadlock

Wait, notify and notifyAll methods

Producer & Consumer problem

Lab 22 & 23:

Create a Deadlock class to demonstrate deadlock in multithreading environment.

Implement wait, notify and notifyAll methods.

Demonstrate how to share threadlocal data between multiple threads.

Session 24:

Lecture:

Inner Class (Regular, Method local, Anonymous & static inner class)

Lambda Expression

Reflection

Lab 24:

Invoke private methods of some other class using reflection.

Create multiple threads using anonymous inner classes.

Create multiple threads using lambda expressions.

Teaching Guidelines for Algorithms and Data Structures Using Java

PG-DAC September 2022

Duration: 72 hours (32 theory hours + 32 lab hours + 8 revision/practice hours)

Objective: To reinforce knowledge of problem solving techniques, data structure concepts and analysis of different algorithms using Java.

Prerequisites: Knowledge of programming in C/C++ with object oriented concepts

Evaluation: 100 Marks

Weightage: Theory exam – 40%, Lab exam – 40%, Internals & Mini project – 20%

Text Book:

- Data Abstraction and Problem Solving with Java: Walls and Mirrors by Janet Prichard , Frank M. Carrano / Pearson

References:

- Problem Solving: Best Strategies to Decision Making, Critical Thinking and Positive Thinking by Thomas Richards / Kindle Edition
 - Data Abstraction and Problem Solving with Java: Walls and Mirrors by Janet Prichard , Frank M. Carrano / Pearson
 - Object-oriented Analysis and Design Using UML - An Introduction to Unified Process and Design Patterns by Mahesh P. Matha / PHI
 - Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein
-

(Note: Each Session is of 2 hours)

Session 1:

Problem Solving & Computational Thinking

Lecture:

- Define the problem
- Identify the problem
- Introduction to Problem Solving
- Problem solving basics

Lab:

- Faculty needs to assign different problems, mostly real world problems
- Students (team-wise, two students in a team) need to analyze as per the techniques learned
- Based on the above problems students need to select as per the selection criteria learned
- They need to implement the selected solution and need to do the documentations.
- Introducing the mini project ideas

Sessions 2 & 3:

Algorithms & Data Structures

Objective: At the end of the session students should know, what is the importance of data structure in problem solving. How stacks, queues, circular queues work. Their real world applications. How to solve problems using these data structures.

Lecture:

- Introductory Concepts
- Algorithm Constructs
- Complexity analysis of algorithms (Big O notation)
- O O design: Abstract Data Types (ADTs)
- Basic Data Structures
 - Arrays
 - Stacks
 - Queues
 - Circular Queues

Lab:

- Implement stack through array
- Complexity analysis of loops and recursive algorithms.
- Implement queues with inserting element at different location (First, Last)
- Implement circular queue

Sessions 4 & 5:

Linked List Data Structures

Objective: At the end of the session students should know, what are applications of Linked List, different types of link list. Comparison with arrays as when to use linked list and when to use array.

Lecture:

- Linked lists
 - Singly linked lists
 - Doubly linked lists
 - Circular linked lists
 - Node-based storage with arrays

Lab:

- Implement circular queue using linked list
- Implement stack using linked list

Session 6:

Recursion

Objective: At the end of the session students should know what is recursion, type of recursion, local variable in recursion, stack manipulation during recursion, function complexity

Lecture & Lab:

- What is recursion?
- What is the base condition in recursion.
- Direct and indirect recursion.
- Memory is allocated to different function calls in recursion.
- Pro and cons of recursion
- Function complexity during recursion

Sessions 7 & 8:

Trees & Applications

Objective: At the end of the session students should know what is the use of binary trees, how to create binary search trees. Different tree traversals. What are the applications of binary trees? How to calculate search complexity in binary search trees? What are the limitations of binary search trees? What are different options to overcome the binary search tree limitations.

Lecture:

- Introduction to trees
- Trees and terminology
- Tree traversals
- Binary trees
- Complete binary trees / Almost complete binary tree (ACBT)
- Array implementation of ACBT
- Binary search trees
- AVL tree

Lab:

- Write a program to implement a binary search tree and the following operations on it:
 - Create()
 - Tree traversals - Breadth First Search, Depth First Search, Inorder(), Preorder(), Postorder()
 - Delete()

Sessions 9, 10 & 11:

Searching & Sorting Algorithms

Objective: At the end of the session students should know what are the different types of sorting and searching algorithms, why all the sorting algorithms are equally important despite different time/space complexity of the algorithms. How the complexity is calculated for each of them. How to pick a sorting algorithm given the nature of the data to be sorted.

Lecture:

- Objectives of Searching
 - The Sequential Search
 - Analysis of Sequential Search
 - The Binary Search
- Analysis of Binary Search
- Introduction to sorting
 - Selection sort
 - Insertion sort
 - Bubble sort
 - Heapsort
 - Mergesort
 - Quicksort
- Analysis of sorting algorithms

Lab:

- Writing program to search an item through sequential search technique.
- Implement to find an item in a list through binary search
- Implement sorting algorithm for: insertion sort, Quicksort

Session 12:

Hash Functions and Hash Tables

Objective: At the end of the session students should know what is hashing, what is the importance of hashing, comparative complexity of hashing with other search techniques. Problems (collision) with hashing and what are the different solutions of that.

Lecture:

- Hashing & Introduction to hash tables
- Hash Functions
- Different type of hash functions

- Collision Resolution
- Linear Probing
- Quadratic Probing
- Double Hashing
- Inserting and Deleting an element from a hash table

Lab:

- Implement hashing techniques in different programs solved earlier
- Write a program to implement Hash table
- Fibonacci recursive algorithm improvement using hash table

Sessions 13 & 14:

Graphs & Applications

Objective: At the end of the session students should know what is graph? Why is graph the most generic data structure? Different types of graphs. Different representation of graph? Graph traversals (Breadth First Traversal, Depth First Traversal). Different applications which can be solved with graphs, real world and programming problems with graphs.

Lecture:

- Introduction to graph theory
- Graph Terminology
- Different types of Graphs
- Representation of Graphs
 - Adjacency Matrix
 - Adjacency List
 - Graph Traversal Algorithms (Breadth First Search, Depth First Search)
- Shortest Path
 - Level Setting : Dijkstra's algorithm
 - Level Correcting: All-pairs shortest path, Floyd-Warshall algorithm
- Spanning Trees
 - Minimum spanning tree algorithms,
 - Prim's algorithm
 - Kruskal's Algorithm

Lab:

- Implement a graph using adjacency Matrix and traverse using Depth First Search.
- Implement a graph and do traversal using stack and queue.

Sessions 15 & 16:

Algorithm Designs

Objective: At the end of the session students should know what are different classes of algorithms. What is the nature of each class of algorithms? How to pick an algorithm for a particular problem. What problems fall under each class of algorithms. What are the worst case, average case and the best case for algorithms?

Lecture:

- What are the different class of algorithms
- How to write efficient Algorithm
- Introduction to algorithm design techniques
- Algorithm Design techniques
- Analysis of an Algorithm
 - Asymptotic Analysis
 - Algorithm Analysis

- Analysis of different type of Algorithms
 - Divide and Conquer Algorithm
 - Greedy algorithm
 - Dynamic Programming algorithm
 - Brute force algorithm
 - Backtracking algorithms
 - Branch-and-bound algorithms
 - Stochastic algorithms
- Complexity
 - Complexity Analysis
 - Space complexity of algorithm
 - Time complexity of algorithm
- Case study on Algorithm Design techniques
- Application of Data structures

Lab + Assignment:

- Study on different Algorithms
- Compare different Algorithms previously programmed and do the analysis

Teaching Guidelines for Database Technologies

PG-DAC September 2022

Duration: 72 hours (32 theory hours + 32 lab hours + 8 revision/practice hours)

Objective: To introduce students to RDBMS and NoSQL Databases and facilitate hands-on experience on SQL (using MySQL) and MongoDB.

Prerequisites: Working knowledge of Windows and Linux, familiarity with programming.

Evaluation: 100 Marks

Weightage: Theory Exam – 40%, Lab exam – 40%, internals – 20%

Text Book:

- Murach's MySQL by Joel Murach / Shroff Publisher

References:

- Database System Concepts by Abraham Silberschatz, Henry Korth and S. Sudarshan / McGraw Hill
 - Database Design and Relational Theory: Normal Forms and All That Jazz by C. J. Date (Author) / O'Reilly
 - Fundamentals of Database System by Shamkant B. Navathe, Ramez Elmasri / Pearson
 - MySQL: The Complete Reference by Vikram Vaswani / McGraw Hill
 - SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management by Andreas Meier and Michael Kaufmann / Springer
 - MongoDB: The Definitive Guide by Shannon Bradshaw, Eoin Brazil and Kristina Chodorow / O'Reilly
 - <http://bigdata.stratebi.com/?language=en>
-

(Note: Each Lecture and Lab Session is of 2 hours)

Session 1:

Lecture

Introduction to DBMS, Basic Database Terminology

Types of DBMS: Relational, Object Relational and NoSQL Databases

Introduction to MySQL, MySQL Clients (Monitor, Shell, Workbench)

Lab

Using MySQL Monitor, Shell, and Workbench

Session 2:

Lecture

Data Models (Conceptual, Logical, Physical)

Database Design, Entity-Relationship Diagram (ERD)

Codd's 12 rules for RDBMS

Introduction to SQL, Categories of SQL Commands: DDL, DML, DCL, DTL/TCL

DDL (CREATE/ALTER/DROP/TRUNCATE)

Lab

Performing basic CREATE, ALTER, DROP Commands

Session 3:**Lecture**

Data Redundancy, Data Anomalies, Functional Dependency

Normalization, Need for Normalization

Normal Forms (1st NF, 2nd NF, 3rd NF, BCNF) with examples, Introduction to 4th and 5th NF

DML (INSERT/UPDATE/DELETE)

Lab

DML (INSERT/UPDATE/DELETE), TRUNCATE

Session 4:**Lecture**

MySQL Data Types, Database Constraints (Primary Key, Unique, Not Null, Foreign Key, Default, Check*)

Aggregate Functions, Grouping Things Together (Group By, Having)

LIKE Operator, DISTINCT, Sorting (Order by clause)

BETWEEN... AND Operators, Comparing Nulls (IS NULL/IS Not NULL), IN/NOT IN

Lab

Defining Data Types for Columns

Creating, Altering, Dropping Constraints

Aggregate Functions: SUM(), AVG(), COUNT(), MAX(), MIN(), COUNT(), Group By, Having Clause

Using Like, Distinct, Order By, Between...And

Comparing Nulls, Using IN/Not-In

Session 5:**Lecture**

Relational Algebra Operations (Selection, Projection, Union, Intersect*, Minus*, Cross/Cartesian)

Joins (Eqvi, Inner, Outer, Natural, Cross), SQL Standard Syntax for Joins

Copying table structure/data, Sequences (AUTO_INCREMENT)

Lab

Union/Union ALL

Queries on Various type of Joins using OLD and SQL Standard Syntax

Copying table structure, Copying data from one table to another

Using AUTO_INCREMENT

Session 6:**Lecture**

Subquery, Correlated Subquery, EXISTS/NOT EXISTS

TCL Commands (Commit/Rollback/Savepoint), DCL Commands (GRANT/REVOKE/GRANT OPTION)

Views, Types of Views, Simple and Complex Views

Lab

Subqueries, Correlated Queries

Using Exists/Not-Exists

Using Commit/Rollback/Savepoint

Granting/revoking privileges on database objects

Creating Views, Querying using Views

Creating Indexes

Creating Temporary Tables

Session 7:**Lecture**

Indexes, Benefit of Indexes, Type of Indexes, Temporary Tables

ACID Properties, Concept of Database Instance and Schema

MySQL Storage Engines (InnoDB, MyISAM and others),

Lab

Indexes, Temporary Tables

All other SQL Commands Revision

Session 8:**Lecture**

Introduction to MySQL Programming, Use of MySQL Programs,

Introduction to Stored Procedures, Benefits of Stored Procedures

Procedure Parameters (IN, OUT and INOUT).

Lab

Creating procedure without parameters

Creating Procedure with (IN/OUT/INOUT) Parameters

Session 9:**Lecture**

Flow Control Statements (LOOP, WHILE and REPEAT)

Using above statements in Stored Procedures/Functions

Conditional Statements (IF, IF-ELSE-THEN, SWITCH CASE)

Example of each type of statement

Lab

Use of flow control statement in Stored Procedure

Use of conditional statements in Stored Procedure

Session 10:**Lecture**

Loop constructs (ITERATE, LEAVE)

Functions with and without parameters

MySQL Built-in functions (string, numeric, date etc.)

Lab

Creating Function and returning value from it

Use of built-in functions in queries

Session 11:**Lecture**

Cursors (Asensitive, Insensitive, Read only, Nonscrollable)

Cursors example and real time use

Lab:

Writing procedures with Declare, fetch and close cursor

Example of each type of cursors

Session 12:**Lecture**

Triggers (BEFORE, AFTER), New and Old trigger variables

Trigger Examples and real time use

Lab

Create Before Triggers

Create After Triggers

Session 13:**Lecture**

Error Handling and Exceptions, Types of Handler Actions, How to write Handler

Defining and handling exceptions in Stored Procedures and Functions

Lab

Exception handling in Stored Procedure

Exception handling with various handler actions

Session 14:**Lecture**

Introduction to NoSQL database, Features of NoSQL Database

Structured vs. Semi-structured and Unstructured Data

Difference between RDBMS and NoSQL databases

CAP Theorem, BASE Model

Categories of NoSQL Databases: Key-Value Store, Document Store, Column-Oriented, Graph

Introduction to MongoDB, Features of MongoDB

MongoDB command interface and MongoDB compass

Lab

Using MongoDB Shell and Compass

Session 15 & 16:**Lecture**

MongoDB Documents & Collections

RDBMS & MongoDB analogies: relations/tables => collections; tuples/records => documents

JSON and BSON documents

Performing CRUD (CREATE, READ, UPDATE, DELETE) Operations, UPSERT

MongoDB – Operators, Sorting, Indexing

Lab:

Creating database, Connecting to a database, Creating Collections

Performing CRUD operations

MongoDB: Complex Read Using Operators, Sorting Operations, Creating Indexes

Teaching Guidelines for Software Development Methodologies

PG-DAC September 2022

Duration: 80 hours (38 theory hours + 34 lab hours + 8 revision/practice hours)

Objective: To build knowledge of Software development methodologies.

Evaluation: 100 marks

Weightage: Theory exam – 40%, Lab exam – 40%, Internals – 20%

Text Book:

- Software Engineering by Chandramouli / Pearson

References:

- Software engineering by Ian Sommerville / Pearson
- Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin / Prentice Hall
- The Mythical Man-Month: Essays on Software Engineering by Frederick P. Brooks Jr. / Addison Wesley
- User Stories Applied: For Agile Software Development by Mike Cohn / Addison Wesley
- DevOps: Continuous Delivery, Integration, and Deployment with DevOps by Sricharan Vadapalli / Packt
- Git for Teams by Emma Westby / O'Reilly

(Note: Each Session is of 2 hours)

Git (8 hours)

Sessions 1 & 2

Lecture

- Developing an application in a team
- Issues developers face when working in a team
- Introduction to code versioning system
- History of code versioning system
 - Different tools available for versioning
 - Software development workflow
- Introduction to git
- Introduction to git repository and git structure
- Adding code to git
- Creating and merging different git branches

Lab

- Create a local git repository
- Commit the initial code
- Update the code
- Use git commands to
 - Get the updated files
 - List the changes
 - Create branch
 - Merge branch

Software Engineering (18 hours)

Sessions 3, 4 & 5

Lecture

- Introduction to software engineering
 - Software Process
 - Software Process Model
 - Software Product
- Importance of Software engineering
- Software Development Life Cycles
- Requirements Engineering
 - Types of Requirements
 - Steps involved in Requirements Engineering
 - Requirement Analysis Modelling
- Design and Architectural Engineering
 - Characteristics of Good Design
 - Function Oriented vs Object Oriented System
 - Modularity, Cohesion, Coupling, Layering
 - Design Models
 - UML
- Coding
 - Programming Principles
 - Coding Conventions
- Object Oriented Analysis and Design

Lab (4 hours)

- Prepare software requirement specification for the final project
- Create the initial use-cases, activity diagram and ER diagram for the final project

Sessions 6 & 7

Lecture

- Introduction to Agile development model
- Agile development components
- Benefits of Agile
- Introduction to different tools used for agile web development
- Scrum and Extreme Programming
- Introduction to Atlassian Jira
 - Add Project
 - Add Tasks and sub-tasks
 - Create sprints with tasks
- Case study of developing web application using agile methodology

Lab

- Create different sprints in Atlassian Jira for different features

DevOps (20 hours)

Sessions 8 & 9

Lecture

- Introduction to Microservices
- Microservices Architecture
- Fragmentation of business requirement
- Deployment pattern

- API gateway
- Service Discovery
- Database Management for Microservices

Lab

- Create Microservices

Sessions 10 & 11**Lecture**

- Introduction to DevOps
- DevOps ecosystem
- DevOps phases
- Introduction to containerisation
- Introduction to docker
- Creating docker images using Dockerfile
- Container life cycle

Lab

- Install and configure docker
- Create docker image using Dockerfile
- Start docker container
- Connect to docker container
- Copy the website code to the container
- Use docker management commands to
 - List the images
 - List the containers
 - Start and stop container
 - Remove container and image

Session 12**Lecture**

- Introduction to YAML
- Introduction to Docker Swarm and Docker Stack
- Introduction to Kubernetes
- Creating Kubernetes cluster
- Creating service in Kubernetes
- Deploying an application using dashboard

Lab

- Configure Kubernetes
- Configure Kubernetes Dashboard
- Setup a Kubernetes cluster
- Access application using Kubernetes service
- Deploy the website using Dashboard

Testing & Integration (18 hours)**Session 13****Lecture**

- Introduction to software testing
- Why testing code is important
- Verification and validation
- Quality Assurance vs Quality Control vs Testing

- Principles of software testing

Assignment

- Read more testing concepts used in the industry

Session 14**Lecture**

- Introduction to STLC and V Model
- Types of testing: manual and automation
- Tools used for automation testing
- Introduction to testing methods: white-box, black-box and grey-box
- Introduction to functional testing: (* students are supposed to learn the concepts)
- Introduction to non-functional testing: (* students are supposed to learn the concepts)

Lab

- Create a test plan for project
- Document the use cases
- Create test case document for different sprints (designed in SE)

Sessions 15 & 16**Lecture**

- Introduction to Selenium (use Eclipse IDE)
- Load web driver
- Create selense commands: locators: by ID, name, class, tag name, XPath
- Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select

Lab

- Download and configure Selenium
- Create a test suite
- Add commands and interactions

Session 17**Lecture**

- Introduction to delivery pipeline
- Introduction to Jenkins
- Jenkins management
- Adding slave node to Jenkins
- Building a delivery pipeline
- Selenium integration with Jenkins

Lab

- Install and configure Jenkins
- Build a pipeline job using Jenkins
- Create a maven project for Selenium
- Add Selenium test suite in the project
- Integrate it with Jenkins

Cloud (8 hours)**Session 18****Lecture**

- Introduction to Cloud
- Introduction to Virtualization
- Virtualization types: type1, type2

- Cloud Computing, Cloud SPI Model, Cloud Computing Types (Public, Private and Hybrid), Cloud Security (SLA and IAM).
- Virtualization, Hardware Virtualization, Para-Virtualization, Cloning, Snapshot and Template
- Containerization, Operating System Virtualization

Lab

- Create and configure VM using VBox
- Deploy code on VM

Session 19**Lecture**

- Cloud architecture
- Service models: IaaS, PaaS, SaaS
- Deployment models: Private, Public, Hybrid
- Services provided by Cloud (Compute, Database, Developer Tools, Storage, Media, Mobile, Web, Security, Integration etc.)
- Cloud development best practices
- Introduction to AWS
- Services provided by AWS: EC2, Lambda, S3

Lab

- Create AWS EC2 instance
 - Add Storage, Tag Instance, Review Instance Launch
 - Set up an Apache web server on your EC2 instance
 - Clean up your EC2 Instance

Teaching Guidelines for Web-based Java Programming

PG-DAC September 2022

Duration: 104 hours (48 theory hours + 46 lab hours + 10 revision/practice hours)

Objective: To learn advanced concepts in java programming and perform web Programming using Java.

Prerequisites: Knowledge of core Java programming

Evaluation: 100 marks

Weightage: Theory exam – 40%, Lab exam – 40%, Internals – 20%

Text Book:

- Core and Advanced Java Black Book / Dreamtech Press

References:

- Servlet and JSP: A Tutorial by Budi Kurniawan / Brainy Software
 - Spring in Action by Craig Walls / Manning Publications
 - Advanced Java programming by Uttam K Roy / Oxford University press
 - Sun Certified Enterprise Architect for Java EE Study Guide by Mark Cade & Humphrey Sheil / Pearson Education
 - Professional Java EE Design Patterns by Murat Yener, Alex Theedom & Reza Rahman / Wrox
-

(Note: Each Session is of 2 hours)

Session 1:

Lecture:

J2EE Overview

- J2EE Container
- Packaging Web applications
- J2EE compliant web application
- Deployment tools.
- Web application life cycle
- Deploying web applications.
- Web Services Support

No Lab

Sessions 2, 3 & 4:

Lecture:

- Servlets: Dynamic Content Generation
- Advantages of Servlets over CGI
- Servlet Life cycle
- Servlet API & Deployment
- Servlet Annotations
- The Servlet interface

- The HttpServlet, HttpServletRequest, HttpServletResponse
- Exception Handling
- Servlet, DAO, POJO DB Layers
- Session
- Session Management
- Session Tracking with
 - Cookies
 - HttpSession
- Request Dispatcher
- Page Navigation
- Complete Case study Servlet Based

Lab:

- Installing a servlet container (Tomcat)
- Adding Server to IDE
- Develop a structured dynamic web application(e.g. Library Management System) using servlets, deploy it in Tomcat
- Use HTTP Session in the Air Ticket Reservation System

Reading: Know more about the HTTP protocol at www.w3c.org

Tutorial: Compare which way of session tracking is better Cookies or HttpSession.

Sessions 5 & 6:**Lecture**

- JSP: Separating UI from Content generation code
- MVC architecture
- Design Pattern: MVC Pattern
- Life cycle of a JSP page
- Directives, Implicit and Explicit Objects, Scriptlets, Expressions, Expression Language
- Scope
- JSP Error Page handling
- JSTL

Lab:

- Separate UI code from the controller code in your Library Management System by incorporating JSP and Servlets.
- Complete the implementation of Air Ticket Reservation System.
- Implement MVC based web application using Servlet, JSP

Sessions 7 & 8:**Lecture:****JDBC & Transaction Management**

- Introduction to JDBC API
- JDBC Architecture
- JDBC Drivers
- JDBC Classes& Interfaces: Driver, Connection, Statement, PreparedStatement, ResultSet and their relationship to provider implementations
- Stored procedures and functions Invocation
- SQL Injection overview and prevention

- Design Pattern: Data Access Object Pattern

Lab:

- Add Database CRUD operations to above MVC web application using JDBC Classes and interfaces. Use DAO and POJO Layers

Sessions 9, 10, 11 & 13:**Lecture:**

- Hibernate Framework
 - Introduction to Hibernate Framework
 - Architecture
- Hibernate in IDE
 - Creating web application using Hibernate API
 - Lifecycle of Hibernate Entities
- HB with annotation example
- Hibernate Mappings and Relationships
- Collection and Component Mapping
- HQL, Named Queries, Criteria Queries

Lab:

- Demonstrate Hibernate as standalone library in Java application
- Develop a web application (Online Bookshop) using Hibernate Persistence

Reading: Study Hibernate architecture from www.hibernate.org/docs

Sessions 13, 14 & 15:**Lecture:**

- What is Spring Framework
- Overview of Spring Architecture
- Spring MVC architecture
- Spring Modules Overview
- Understanding Spring 4 annotations (Basic Introduction)
- What is IoC (Inversion of Control)
- IOC container
- Dependency Injection
- Spring Beans
- Autowiring Beans
- Bean Scopes
- Spring MVC
- Model, Model & View, HandlerMapping, ViewResolver
- Design Pattern: Front Controller Pattern
- Spring MVC Web application with JSP views (without Spring Boot)
- Using Thymeleaf as alternate View Technology (only introduction)
- Spring Validations
- Spring i18n, Localization, Properties
- File Upload example

Lab:

- Design and deploy Library Management System using Spring Web

Sessions 16 & 17:**Lecture:**

- Spring Boot essentials
- Why Spring boot
- Spring Boot Overview
- Basic Introduction of MAVEN
- Building Spring Web application with Boot
- Spring Boot in detail (Use Spring Boot for all demo & assignments here onwards)
- Running a web application using Spring Boot with CRUD (with Static Data not DB)

Lab:

- Create Hello World Spring Boot Web application
- Check Libraries imported by Spring Boot
- Create Spring Boot CRUD application with Thymeleaf as View technology

Sessions 18 & 19:**Lecture:**

Spring Data Module

- Spring Data JPA (Repository support for JPA)
- Crud Repository & JPA Repository
- Query methods
- Using custom query (@Query)

Lab:

- Add CRUD operations with Spring JPA etc. to earlier Spring Web application.

Session 20:**Lecture:**

Spring AOP

- AOP Overview
- Spring AOP
- AOP Terminology and annotations: Advice, Join Points, Pointcuts, Aspects

Lab

- Modify earlier Spring MVC application to Log all the requests using AOP

Sessions 21 & 22:**Lecture:**

Building REST services with Spring

- Introduction to web services
- SOAP Vs RESTful web services
- RESTful web service introduction
- Create RESTful web service in java using Spring Boot
- RESTful web service JSON example
- RESTful web service CRUD example
- Using POSTMAN client to invoke REST API's
- REST service invocation using REST Template

Lab:

- Create REST API for Employee Management using Spring Boot
- Invoke it from POSTMAN app

- Invoke it from another Spring Boot Web application using REST Template

Session 23 & 24:**Lecture:****Testing in Spring**

- Testing in Spring
- Unit Testing of Spring MVC Controllers
- Unit Testing of Spring Service Layer
- Integration Testing of Spring MVC Applications: REST API
- Unit Testing Spring MVC Controllers with REST

Securing Web Application with Spring Security

- What is Spring Security
- Spring Security with Spring Boot
- Basic Authentication
- Authentication with User credentials from Database and Authorization
- JWT Authorization

Lab:

- Design & Test Spring Application
- Secure the Spring Web application created in earlier exercise.

Teaching Guidelines for MS.Net Technologies

PG-DAC September 2022

Duration: 84 hours (42 theory hours + 34 lab hours + 8 revision/practice hours)

Objective: To acquire the knowledge of Microsoft.NET 6.

Prerequisites: Students are expected to know any OOP. They should have undergone the Web Programming module which includes HTML, CSS, JavaScript, JSON, and XML. Knowledge of any database is required.

Note: Training will be carried out on .Net 6 using Visual Studio 2022

Evaluation: 100 marks

Weightage: Theory exam – 40%, Lab exam – 40%, Internals – 20%

Text Book:

- Pro C# 10 with .Net 6 - Foundational Principles and Practices in Programming by Andrew Troelsen & Philip Japikse / Apress

References:

- C# 10 and .Net 6 - Modern Cross-Platform Development by Mark J. Price / Packt
-

(Note: Each Session is of 2 hours)

Session 1:

Lecture:

Introduction to the .Net Framework

Intermediate Language (IL)

Assemblies and their structure, EXEs/DLLs

CLR and its functions

- JIT Compilation
- Memory Management
- Garbage Collection
- AppDomain Management
- Memory Management
- CLS, CTS
- Security

NO LAB

Session 2:

Lecture:

.Net Framework, .Net Core, Mono, Xamarin differences

Versions of the Framework

Managed and Unmanaged Code

Introduction to Visual Studio

Using ILDASM

NO LAB

Session 3:

Lecture:

Console Applications and Class Libraries .Net Core

C# Basics

Project References, using

Classes

Data Types in .net and CTS equivalents

Methods

- Method Overloading
- Optional Parameters
- Named Parameters and Positional Parameters
- Using params
- Local functions

Properties

- get, set
- Readonly properties
- Using property accessors to create Readonly property

Constructors

Object Initializer

Destructors

Discussion on IDisposable. To be implemented after interfaces

Lab:

Create a class that has Properties, Fields, Methods, Constructors (Trainer can specify any class of his choice, e.g. Student, Employee, etc)

Session 4:

Lecture:

Static Members of a Class

- Fields
- Methods
- Properties
- Constructors

Static Classes

Static local functions

Inheritance

- Access Specifiers
- Constructors in a hierarchy
- Overloading in derived class
- Hiding, using new
- override
- sealed methods
- Abstract Classes
- Abstract Methods
- Sealed Classes

Lab:

Create multiple classes that use Inheritance based concepts

Session 5:

Lecture:

Interfaces

- Implementing an interface
- Explicitly implementing an interface
- Inheritance in interfaces
- Default interface methods

Operator overloading

Lab:

Create and implement interfaces for the classes created in Lab 4

Implement IDisposable, IComparable

Session 6:

Lecture:

Reference and Value Types

Value Types

- struct
- enum

out and ref

nullable types

nullable reference types

?? and ??=

Working with Arrays (single, multidim, jagged), Array Class members

Indices and ranges

Indexers

Lab:

Lab based on array examples.

Also create an array of the class created in Lab 1.

Session 7:

Lecture:

Generic classes

Generic methods

Generic Constraints

Collections – generic and non-generic

Collection Examples based on ICollection, IList, IDictionary (both generic and non-generic)

Iterating collections using foreach

Using Tuples to pass multiple values to a function

Lab:

Lab based on collection examples.

Also create a collection of the class created in Lab 1.

Session 8:

Lecture:

Delegates

- Calling methods using delegates
- Uses of delegates
- Multicast delegates
- Action, Func, Predicate delegates

Anonymous methods

Lambdas

Lab:

Lab based on delegates examples.

Session 9:**Lecture:**

Error Handling (Exceptions Handling)

- Checked & Unchecked Statements
- The try, catch, finally
- Dos & Don'ts of Exception Handling

User Defined Exception classes

Declaring and raising events

Handling events

Lab:

Lab based on exceptions and events examples.

Session 10:**Lecture:**

Anonymous types

Extension methods

Partial classes

Partial methods

LINQ to objects

Writing LINQ queries

Deferred execution

LINQ methods

PLINQ

Lab:

Lab based on LINQ examples

Students to try tutorial for 101 LINQ Queries

Session 11:**Lecture:**

Creating a shared assembly

Creating Custom Attributes

Using Reflection to explore an Assembly

Using Reflection to load an Assembly dynamically

Files I/O and Streams

- Working with drivers, Directories, and Files
- Reading and Writing files

NO LAB**Session 12:****Lecture:**

Threading

- ThreadStart, Parameterized ThreadStart
- ThreadPool
- Synchronizing critical data using lock, Monitor and Interlocked

Working with Tasks

- Calling functions with and without return values
- Using async, await

Using the Task Parallel Library

Lab:

Threading related examples

Task related examples

Sessions 13-19:**Lecture:****Introduction to Asp.Net MVC CORE**

- Architecture of an ASP .Net MVC application
- Understanding Folder structures and configuration files

Understanding Controllers and Action

- Create a controller
- How actions are invoked
- `HttpGet` , `HttpPost` , `NoAction` Attributes
- Running Action result.

Understanding Views & Models

- Creating Models & ViewModel
- Creating Razor Views
- HTML Helper Functions
- Understanding ViewBag
- Create a view using ViewBag
- Validation using Data Annotations
- Client side and server side validation
- Self validated model
- Creating Strongly Types Views
- Using Various Scaffold Templates
- CRUD operation using Model

MVC State Management

- ViewBag , TempData , Session , Application
- Cookies , QueryString

MVC Module

- Partial View
- Action Method and child action

Data Management with ADO.NET

- Microsoft.Data.SqlClient introduction
- Connection object, Command object, DataReader, DataAdapter, DataSet and DataTable.
- Asynchronous command Execution
- Asynchronous Connections

Understanding Routing & Request Life Cycle

- Routing Engine & Routing Table
- Understanding and configuring Routing Pattern in RouteConfig File
- Understanding 404 error and resource not found.
- Using Attributes Routing
- Understanding Request Life Cycle

Layouts , Bundle , Minification

- Creating Layout and using with associated views
- Understanding Bundling and Minification
- Using BundleConfig file
- Attaching css , js , bootstrap in bundles
- Custom Helper Function
- Asynchronous Actions
- Error Handling in MVC with Log Entry
- Filters and Custom Action Filter

MVC Security

- Using Authorize & Allow Anonymous attributes

- Implementing Forms Based Authentication
- Preventing Forgery Attack using AntiForgeryToken
- Preventing Cross Site Scripting Attack

Entity Framework

- Introduction to EF
- Different Approaches
- Using Code First Approach
- Using various Data Annotations
- Using Validation, Primary Key , Foreign Key etc
- Using Fluent APIs
- Database Migrations
- CRUD operation using EF

Developing MVC application using EF Code First Approach

Introduction to Razor Pages

Lab:

Lab exercise covering the concepts covered in the class

Session 20:

Lecture:

Localization in MVC (Demo Only)

Deploying ASP .NET MVC application (Demo only)

NO LAB

Session 21

Lecture:

Web APIs

- Creating ASP.NET MVC Web API
- Configuring for CORS
- Different Verbs
- Consuming using a client
- Using Newtonsoft APIs

Lab:

Create a RESTful service using WEB API. Create a consumer.

Teaching Guidelines for **Software Project**

PG-DAC September 2022

Duration: 120 hours

Objective: In addition to the specific subject knowledge, the Software Project module attempts to put into practice a number of things that the students have learned during the PG-DAC course, such as:

- Ability to work in a team
- Software development methodology and principles
- Good programming practices
- Technical reporting and presentation.

Prerequisites: Completion of the basic modules on Programming, Data Structures, and Database to start Phase I of the Project.

Evaluation: Grading based on the combined marks obtained in the evaluations of all the 3 phases.

Weightage: Phase I – 10%, Phase II – 10%, Phase III – 80% (Mid Evaluation 20% + Final Evaluation 60%)

Final Project Schedule

Students, in teams, will be required to identify project topics in consultation with faculty members within the first three months of the course.

The Project module is divided in three phases.

I – SRS Phase:

Tasks: Requirements gathering, feasibility study and project thinking.

Deliverable: Software Requirement Specification (SRS).

Schedule: Right after the delivery of the basic modules. This phase will be executed along with the Software Engineering part of the Software Development Methodologies module to enable better absorption of the concepts.

II – Design Phase:

Tasks: Software design and project plan.

Deliverable: Students will present the design and plan on the schema of the project.

Schedule: This phase will be executed during the final part of the Software Development Methodologies module.

III – Development Phase:

Tasks: Coding and testing of the software system/application.

Deliverables: Project report, functional software system/application.

Schedule: This final phase will be executed during the last 15 days of the course. A mid evaluation at the middle of the project development, and a final evaluation at the end of the project will be done.