Course Category: Specialization -I

(Application Development Paradigm)

Semester III

Programming in C/C++

(Mentor: Mrs. Mary Margarat)

1. **Nptel: Introduction to programming in C**
   Link: https://swayam.gov.in/nd1_noc20_cs91/preview
   (8 weeks)

Course layout

Week 1 : introduction. Straight-line code. Variables, operators, expressions and conditionals.
Week 2 : loops
Week 3 : functions
Week 4 : one-dimensional arrays and pointers
Week 5 : recursion
Week 6 : multi-dimensional arrays, linked lists.
Week 7 : operating on files
Week 8 : organizing c projects, working with multiple source directories, makefiles.

2. Coursera: Introduction to programming in C

Link: Introduction To Programming in C

There are 4 Courses in this Specialization

C O U R S E 1:

Programming Fundamentals

Programming is an increasingly important skill, whether you aspire to a career in software development, or in other fields. This course is the first in the specialization Introduction to Programming in C, but its lessons extend to any language you might want to learn. This is because programming is fundamentally about figuring out how to solve a class of problems and writing the algorithm, a clear set of steps to solve any problem in its class. This course will introduce you to a powerful problem-solving process—the Seven Steps—which you can use to solve any programming problem. In this course, you will learn how to develop an algorithm, then progress to reading code and understanding how programming concepts relate to algorithms.

C O U R S E 2:

Writing, Running, and Fixing Code in C

Building on the course Programming Fundamentals, this course will teach you how to write code by first planning what your program should do—an important approach for novice and professional programmers. You will learn how to compile and run your program, and then how to test and debug it. This course builds on the Seven Steps you have already learned and provides a framework for systematically testing for problems and fixing them, so you can find and fix problems efficiently.

C O U R S E 3:

Pointers, Arrays, and Recursion

The third course in the specialization Introduction to Programming in C introduces the programming constructs pointers, arrays, and recursion. Pointers provide control and flexibility when programming in C by giving you a way to refer to the location of other data. Arrays provide a way to bundle data by guaranteeing sequences of data are grouped together. Finally, recursive functions—functions that call themselves—provide an alternative to iteration that are very useful for implementing certain algorithms.

C O U R S E 4:

Interacting with the System and Managing Memory

The final course in the specialization Introduction to Programming in C will teach you powerful new programming techniques for interacting with the user and the system and dynamically allocating memory. You will learn more sophisticated uses for pointers, such as strings and multidimensional arrays, as well as how to write programs that read and write files and take input from the user. Learning about dynamic memory allocation will allow your programs to perform complex tasks that will be applied in the final part of the specialization project: a Monte Carlo simulation for calculating poker hand probabilities.

Semester IV

Introduction to programming & Data structure algorithms

(Mentor: Mr Bhushan Nemade)

1. Nptel: Programming, Data Structures And Algorithms Using Python
   Link: Https://swayam.gov.in/nd1_noc20_cs70/preview (8 weeks)

C O U R S E L A Y O U T

Week 1

Informal introduction to programmin, algorithms and data structures viagcd

Downloading and installing Python

gcd in Python: variables, operations, control flow - assignments, condition-als, loops, functions

Week 2

Python: types, expressions, strings, lists, tuples

Python memory model: names, mutable and immutable values

List operations: slices etc

Binary search

Inductive function definitions: numerical and structural induction

Elementary inductive sorting: selection and insertion sort

In-place sorting

Week 3

Basic algorithmic analysis: input size, asymptotic complexity, O() notation

Arrays vs lists

Merge sort

Quicksort

Stable sorting

Week 4

Dictionaries

More on Python functions: optional arguments, default values

Passing functions as arguments

Higher order functions on lists: map, list comprehension

Week 5

Exception handling

Basic input/output

Handling files

String processing

Week 6

Backtracking: N Queens, recording all solutions

Scope in Python: local, global, nonlocal names

Nested functions

Data structures: stack, queue

Heaps

Week 7

Abstract datatypes

Classes and objects in Python

"Linked" lists: find, insert, delete

Binary search trees: find, insert, delete

Height-balanced binary search trees

Week 8

Effcient evaluation of recursive definitions: memoization

Dynamic programming: examples

Other programming languages: C and manual memory management

Other programming paradigms: functional programming

2. **Unordered Data Structure: Coursera**
   Link: https://www.coursera.org/learn/cs-fundamentals-3

The Unordered Data Structures course covers the data structures and algorithms needed to implement hash tables, disjoint sets and graphs. These fundamental data structures are useful for unordered data. For example, a hash table provides immediate access to data indexed by an arbitrary key value, that could be a number (such as a memory address for cached memory), a URL (such as for a web cache) or a dictionary. Graphs are used to represent relationships between items, and this course covers several different data structures for representing graphs and several different algorithms for traversing graphs, including finding the shortest route from one node to another node. These graph algorithms will also depend on another concept called disjoint sets, so this course will also cover its data structure and associated algorithms.

W E E K 1:Orientation; Hashing
W E E K 2:Disjoint Sets
W E E K 3:Graph Data Structures
W E E K 4:Graph Algorithms

3. Data structures : Coursera

Link: https://www.coursera.org/learn/data-structures

A good algorithm usually comes together with a set of good data structures that allow the algorithm to manipulate the data efficiently. In this course, we consider the common data structures that are used in various computational problems. You will learn how these data structures are implemented in different programming languages and will practice implementing them in our programming assignments. This will help you to understand what is going on inside a particular built-in implementation of a data structure and what to expect from it. You will also learn typical use cases for these data structures.

A few examples of questions that we are going to cover in this class are the following: 1. What is a good strategy of resizing a dynamic array? 2. How priority queues are implemented in C++, Java, and Python? 3. How to implement a hash table so that the amortized running time of all operations is O(1) on average? 4. What are good strategies to keep a binary tree balanced? You will also learn how services like Dropbox manage to upload some large files instantly and to save a lot of storage space!

W E E K 1:Basic Data Structures

In this module, you will learn about the basic data structures used throughout the rest of this course. We start this module by looking in detail at the fundamental building blocks: arrays and linked lists. From there, we build up two important data structures: stacks and queues. Next, we look at trees: examples of how they’re used in Computer Science, how they’re implemented, and the various ways they can be traversed. Once you’ve completed this module, you will be able to implement any of these data structures, as well as have a solid understanding of the costs of the operations, as well as the tradeoffs involved in using each data structure.

W E E K 2:Dynamic Arrays and Amortized Analysis

In this module, we discuss Dynamic Arrays: a way of using arrays when it is unknown ahead-of-time how many elements will be needed. Here, we also discuss amortized analysis: a method of determining the amortized cost of an operation over a sequence of operations. Amortized analysis is very often used to analyse performance of algorithms when the straightforward analysis produces unsatisfactory results, but amortized analysis helps to show that the algorithm is actually efficient. It is used both for Dynamic Arrays analysis and will also be used in the end of this course to analyze Splay trees.

W E E K 3:Priority Queues and Disjoint Sets

We start this module by considering priority queues which are used to efficiently schedule jobs, either in the context of a computer operating system or in real life, to sort huge files, which is the most important building block for any Big Data processing algorithm, and to efficiently compute shortest paths in graphs, which is a topic we will cover in our next course. For this reason, priority queues have built-in implementations in many programming languages, including C++, Java, and Python. We will see that these implementations are based on a beautiful idea of storing a complete binary tree in an array that allows to implement all priority queue methods in just few lines of code. We will then switch to disjoint sets data structure that is used, for example, in dynamic graph connectivity and image processing. We will see again how simple and natural ideas lead to an implementation that is both easy to code and very efficient. By completing this module, you will be able to implement both these data structures efficiently from scratch.

W E E K 4:Hash Tables

In this module you will learn about very powerful and widely used technique called hashing. Its applications include implementation of programming languages, file systems, pattern search, distributed key-value storage and many more. You will learn how to implement data structures to store and modify sets of objects and mappings from one type of objects to another one. You will see that naive implementations either consume huge amount of memory or are slow, and then you will learn to implement hash tables that use linear memory and work in O(1) on average! In the end, you will learn how hash functions are used in modern disrtibuted systems and how they are used to optimize storage of services like Dropbox, Google Drive and Yandex Disk!

4. **Python Data structures: courser**

   Link: https://www.coursera.org/learn/python-data

scourse will introduce the core data structures of the Python programming language. We will move past the ics of procedural programming and explore how we can use the Python built-in data structures such as lists, ionaries, and tuples to perform increasingly complex data analysis. This course will cover Chapters 6-10 of textbook “Python for Everybody”. This course covers Python 3.

WEEK 1:Chapter Six: Strings

In this class, we pick up where we left off in the previous class, starting in Chapter 6 of the textbook and covering Strings and moving into data structures. The second week of this class is dedicated to getting Python installed if you want to actually run the applications on your desktop or laptop. If you choose not to install Python, you can just skip to the third week and get a head start.

W E E K 2:Unit: Installing and Using Python

In this module you will set things up so you can write Python programs. We do not require installation of Python for this class. You can write and test Python programs in the browser using the "Python Code Playground" in this lesson. Please read the "Using Python in this Class" material for details.

W E E K 3 : Chapter Seven: Files

Up to now, we have been working with data that is read from the user or data in constants. But real programs process much larger amounts of data by reading and writing files on the secondary storage on your computer. In this chapter we start to write our first programs that read, scan, and process real data.

W E E K 4: Chapter Eight: Lists

As we want to solve more complex problems in Python, we need more powerful variables. Up to now we have been using simple variables to store numbers or strings where we have a single value in a variable. Starting with lists we will store many values in a single variable using an indexing scheme to store, organize, and retrieve different values from within a single variable. We call these multi-valued variables "collections" or "data structures".

5. **Algorithms, Part-I: courser**

   Link: https://www.coursera.org/learn/algorithms-part1

This course covers the essential information that every serious programmer needs to know about algorithms and data structures, with emphasis on applications and scientific performance analysis of Java implementations. Part I covers elementary data structures, sorting, and searching algorithms. Part II focuses on graph- and string-processing algorithms.

All the features of this course are available for free. It does not offer a certificate upon completion.

W E E K 1: Course Introduction

Welcome to Algorithms, Part I.

Union−Find

We illustrate our basic approach to developing and analyzing algorithms by considering the dynamic connectivity problem. We introduce the union−find data type and consider several implementations (quick find, quick union, weighted quick union, and weighted quick union with path compression). Finally, we apply the union−find data type to the percolation problem from physical chemistry.

Analysis of Algorithms

The basis of our approach for analyzing the performance of algorithms is the scientific method. We begin by performing computational experiments to measure the running times of our programs. We use these measurements to develop hypotheses about performance. Next, we create mathematical models to explain their behavior. Finally, we consider analyzing the memory usage of our Java programs.

W E E K 2: Stacks and Queues

We consider two fundamental data types for storing collections of objects: the stack and the queue. We implement each using either a singly-linked list or a resizing array. We introduce two advanced Java features—generics and iterators—that simplify client code. Finally, we consider various applications of stacks and queues ranging from parsing arithmetic expressions to simulating queueing systems.

Elementary Sorts

We introduce the sorting problem and Java's Comparable interface. We study two elementary sorting methods (selection sort and insertion sort) and a variation of one of them (shellsort). We also consider two algorithms for uniformly shuffling an array. We conclude with an application of sorting to computing the convex hull via the Graham scan algorithm.

W E E K 3:Mergesort

We study the mergesort algorithm and show that it guarantees to sort any array of n items with at most n lg n compares. We also consider a nonrecursive, bottom-up version. We prove that any compare-based sorting algorithm must make at least n lg n compares in the worst case. We discuss using different orderings for the objects that we are sorting and the related concept of stability.

Quicksort

We introduce and implement the randomized quicksort algorithm and analyze its performance. We also consider randomized quickselect, a quicksort variant which finds the kth smallest item in linear time. Finally, we consider 3-way quicksort, a variant of quicksort that works especially well in the presence of duplicate keys.

W E E K 4:Priority Queues

We introduce the priority queue data type and an efficient implementation using the binary heap data structure. This implementation also leads to an efficient sorting algorithm known as heapsort. We conclude with an applications of priority queues where we simulate the motion of n particles subject to the laws of elastic collision.

Elementary Symbol Tables

We define an API for symbol tables (also known as associative arrays, maps, or dictionaries) and describe two elementary implementations using a sorted array (binary search) and an unordered list (sequential search). When the keys are Comparable, we define an extended API that includes the additional methods min, max floor, ceiling, rank, and select. To develop an efficient implementation of this API, we study the binary search tree data structure and analyze its performance.

Semester V
Programming with Java/ Object Oriented Programming
(Mentor: Mr. Sudhir Dhekane)

1. **Programming in Java: NPTEL**
   Link: https://swayam.gov.in/nd1_noc20_cs58/preview (12 Weeks)

COURSE LAYOUT

Week 1 : Overview of Object-Oriented Programming and Java

Week 2 : Java Programming Elements
Week 3 : Input-Output Handling in Java
Week 4 : Encapsulation
Week 5 : Inheritance
Week 6 : Exception Handling
Week 7 : Multithreaded Programming
Week 8 : Java Applets and Servlets
Week 9 : Java Swing and Abstract Windowing Toolkit (AWT)
Week 10 : Networking with Java
Week 11: Java Object Database Connectivity (ODBC)
Week 12: Interface and Packages for Software Development

BOOKS AND REFERENCES

1.Java: The Complete Reference Hebert Schildt, Mc Graw Hill
2. Object-Oriented Programming with C++ and Java Debasis Samanta, Prentice Hall India.

2. **Object oriented programming in java specialization: courser**
   Link: https://www.coursera.org/specializations/object-oriented-programming

This Specialization is for aspiring software developers with some programming experience in at least one other programming language (e.g., Python, C, JavaScript, etc.) who want to be able to solve more complex problems through objected-oriented design with Java. In addition to learning Java, you will gain experience with two Java development environments (BlueJ and Eclipse), learn how to program with graphical user interfaces, and learn how to design programs capable of managing large amounts of data. These software engineering skills are broadly applicable across wide array of industries.

There are 4 Courses in this Specialization

C O U R S E 1:

Java Programming: Solving Problems with Software

Learn to code in Java and improve your programming and problem-solving skills. You will learn to design algorithms as well as develop and debug programs. Using custom open-source classes, you will write programs that access and transform images, websites, and other types of data. At the end of the course you will build a program that determines the popularity of different baby names in the US over time by analyzing comma separated value (CSV) files.

After completing this course you will be able to: 1. Edit, compile, and run a Java program; 2. Use conditionals and loops in a Java program; 3. Use Java API documentation in writing programs. 4. Debug a Java program using the scientific method; 5. Write a Java method to solve a specific problem; 6. Develop a set of test cases as part of developing a program; 7. Create a class with multiple methods that work together to solve a problem; and 8. Use divide-and-conquer design techniques for a program that uses multiple methods.

C O U R S E 2:

Java Programming: Arrays, Lists, and Structured Data

Build on the software engineering skills you learned in "Java Programming: Solving Problems with Software" by learning new data structures. Use these data structures to build more complex programs that use Java's object-oriented features. At the end of the course you will write an encryption program and a program to break your encryption algorithm.

After completing this course, you will be able to: 1. Read and write data from/to files; 2. Solve problems involving data files; 3. Perform quantitative analyses of data (e.g., finding maximums, minimums, averages); 4. Store and manipulate data in an array or ArrayList; 5. Combine multiple classes to solve larger problems; 6. Use iterables and collections (including maps) in Java.

C O U R S E 3:

Object Oriented Programming in Java

Welcome to our course on Object Oriented Programming in Java using data visualization. People come to this course with many different goals -- and we are really excited to work with all of you! Some of you want to be professional software developers, others want to improve your programming skills to implement that cool personal project that you've been thinking about, while others of you might not yet know why you're here and are trying to figure out what this course is all about.

This is an intermediate Java course. We recommend this course to learners who have previous experience in software development or a background in computer science. Our goal is that by the end of this course each and every one of you feels empowered to create a Java program that's more advanced than any you have created in the past and that is personally interesting to you. In achieving this goal you will also learn the fundamentals of Object Oriented Programming, how to leverage the power of existing libraries, how to build graphical user interfaces, and how to use some core algorithms for searching and sorting data. And this course is project-based, so we'll dive right into the project immediately! We are excited to be offering a unique course structure, designed to support learners of different backgrounds in succeeding at their own pace. The first module explains how this will work and if this course is right for you. We also recommend taking a few minutes to explore the course site. A good place to start is the navigation bar on the left. Click Course Content to see what material we'll cover each week, as well preview the assignments you'll need to complete to pass the course. Click Discussions to see forums where you can discuss the course material with fellow students taking the class. Be sure to introduce yourself to everyone in the Meet and Greet forum. This course should take about 6 weeks to complete. You can check out the recommended course schedule below to see a quick overview of the lessons and assignments you'll complete each week. We're excited you're here learning with us. Let's get started!

C O U R S E 4:

Data Structures and Performance

How do Java programs deal with vast quantities of data? Many of the data structures and algorithms that work with introductory toy examples break when applications process real, large data sets. Efficiency is critical, but how do we achieve it, and how do we even measure it?

This is an intermediate Java course. We recommend this course to learners who have previous experience in software development or a background in computer science, and in particular, we recommend that you have taken the first course in this specialization (which also requires some previous experience with Java). In this course, you will use and analyze data structures that are used in industry-level applications, such as linked lists, trees, and hashtables. You will explain how these data structures make programs more efficient and flexible. You will apply asymptotic Big-O analysis to describe the performance of algorithms and evaluate which strategy to use for efficient data retrieval, addition of new data, deletion of elements, and/or memory usage. The program you will build throughout this course allows its user to manage, manipulate and reason about large sets of textual data. This is an intermediate Java course, and we will build on your prior knowledge. This course is designed around the same video series as in our first course in this specialization, including explanations of core content, learner videos, student and engineer testimonials, and support videos -- to better allow you to choose your own path through the course!

Semester VI
Web Programming Paradigms
(Mentor: Mr. Shridhar Kamble)

1.Introduction to web development
Link: https://www.coursera.org/learn/web-development

This course is designed to start you on a path toward future studies in web development and design, no matter how little experience or technical knowledge you currently have. The web is a very big place, and if you are the typical internet user, you probably visit several websites every day, whether for business, entertainment or education. But have you ever wondered how these websites actually work? How are they built? How do browsers, computers, and mobile devices interact with the web? What skills are necessary to build a website? With almost 1 billion websites now on the internet, the answers to these questions could be your first step toward a better understanding of the internet and developing a new set of internet skills.

By the end of this course you'll be able to describe the structure and functionality of the world wide web, create dynamic web pages using a combination of HTML, CSS, and JavaScript, apply essential programming language concepts when creating HTML forms, select an appropriate web hosting service, and publish your webpages for the world to see. Finally, you'll be able to develop a working model for creating your own personal or business websites in the future and be fully prepared to take the next step in a more advanced web development or design course or specialization.

W E E K 1:

Course Overview and Website Structure and Hosting

This first module provides an overview of how websites function, their structure, and the ins and outs of choosing a website name and selecting an online host to house your website. By the end of this module, you'll be able to: find and select a web hosting company; choose an effective domain name; use the host to manage your websites; and discuss how networks and the internet function at a high level.

W E E K 2:

Designing Your Own Website: HTML Basics

In this module, we'll begin to explore how to design and create websites by exploring the base language used to power all websites: HTML. By the end of this lesson, you'll be able to: identify and use common HTML tags; add an image to a webpage; create HTML-formatted tables; use hyperlinks to connect a series of webpages; upload your finished HTML pages to a web host; and, learn some tips and tricks for styling pages and practicing your coding.

W E E K 3:

Introduction to Programming Using JavaScript

Now that you know some basic HTML, it's time to turn our attention to another common scripting language used to make websites dynamic - that is allowing users to interact with your webpages - JavaScript. While learning about JavaScript, you'll also gain some foundational knowledge common to all programming languages. By the end of this module, you'll be able to: discuss what is meant by dynamic content; perform essential programming language tasks; create simple JavaScript programs; use JavaScript to set up alerts and respond to events, to read input, and to change HTML; and conduct basic JavaScript testing.

W E E K 4

Websites with Style: CSS Properties, Colors and Fonts

While HTML and JavaScript are very useful for web development, they don't exactly make websites look attractive - that's where cascading style sheets, or CSS, comes into play. While HTML is used to build the structure of our pages and JavaScript is used to provide interactive functionality, CSS is used to graphically design and layout webpages. By the end of this module, you'll be able to: discuss common mistakes in designing a website; identify and apply CSS basics like purpose and syntax; use CSS properties to control fonts, colors, layouts, and other common properties; differentiate between in-line, internal, and external CSS; and practice and test your cascading style sheets.

2. **Responsive website basics: Code with HTML, CSS, Javascript.**
   Link: https://www.coursera.org/learn/website-coding

In this course you will learn three key website programming and design languages: HTML, CSS and JavaScript. You will create a web page using basic elements to control layout and style. Additionally, your web page will support interactivity.

At the end of the course, you will be able to: 1. Define the purpose of HTML, CSS and JavaScript 2. Make a simple web page using HTML 3. Use CSS to control text styles and layout 4. Use CSS libraries such as Bootstrap to create responsive layouts 5. Use JavaScript variables and functions 6. Manipulate web page content using JavaScript 7. Respond to user input using JavaScript In this course, you will complete: 2 assignments writing HTML, CSS and JavaScript, each taking ~1 hour to complete 4 quizzes, each taking ~20 minutes to complete 1 programming exercise~30 minutes to complete multiple practice quizzes, each taking ~5 minutes to complete.

Course introduction

Welcome to the first course of the 'Responsive website development and design' specialisation!

W E E K 1 : HTML

We start the course by looking at how to set up a dev environment, build a HTML navbar and how to embed images and create lists using HTML.<p>Also we'll create properly structured HTML documents and have a look at the world's first web page. <p> Looking forward to working with you this week! <p> Matthew, Marco and Kate

W E E K 2: CSS

Welcome to the second module of 'Responsive website basics'. <p>In this section of the course we will have a look at linking external CSS files to your HTML documents, controlling fonts with CSS and using CSS to customise hyperlink formatting and to control text layout. We will also install the bootstrap library and implement a responsive grid layout- Enjoy! <p> - Matthew, Marco and Kate

W E E K 3: Beginning JavaScript

Welcome to the third module of 'Responsive website basics'. <p>In this module we will write simple JavaScript programs and learn how to write programs that can respond to user input such as clicking on HTML elements. We will also take a look at JavaScript functions and use jQuery to manipulate web pages.</p>Finally, you will learn how to write your own javaScript functions including anonymous functions. <p> Looking forward to working with you this week! <p> - Matthew and Marco

W E E K 4: Going deeper into JavaScript

Welcome to the final module of 'Responsive website basics'. <p>In this final section of the course we will define JavaScript variables and write simple JavaScript programs that use and change the values of variables. </p>We'll also write if statements to control the flow of a JavaScript program, use boolean variables in conjunction with if statements and write javaScript in the context of moderately complex web applications. <p> Enjoy! <p> - Matthew and Marco

3. **Building Web application using Php**
   Link: https://www.coursera.org/learn/web-applications-php

In this course, you'll explore the basic structure of a web application, and how a web browser interacts with a web server. You'll be introduced to the request/response cycle, including GET/POST/Redirect. You'll also gain an introductory understanding of Hypertext Markup Language (HTML), as well as the basic syntax and data structures of the PHP language, variables, logic, iteration, arrays, error handling, and superglobal variables, among other elements. An introduction to Cascading Style Sheets (CSS) will allow you to style markup for webpages. Lastly, you'll gain the skills and knowledge to install and use an integrated PHP/MySQL environment like XAMPP or MAMP.

W E E K 1

Introduction to Dynamic Web Content

We look at the basic structure of a web application and how a web browser interacts with a web server. We explore the Request-Response Cycle that is the basis of the Hypertext Transfer Protocol (HTTP).

W E E K 2

HyperText Markup Language (HTML)

We briefly cover the basics of the HyperText Markup Language (HTML) that is the markup for web pages. We hope that you already have some expertise in HTML and that this is mostly review.

W E E K 3

Cascading Style Sheets (CSS)

We briefly cover the basics of cascading Style Sheets (CSS) that allow us to style the markup for web pages.

W E E K 4

Installing PHP and SQL

Our first technical task is to work through the installation steps including installing a text editor, installing MAMP or XAMPP (or equivalent), creating a MySql Database, and writing a PHP program.

4. Web development
   Link: https://www.coursera.org/learn/web-application-development?specialization=website-development

In this course, you will develop more advanced web application programming skills. You will learn how to control data read and write access using methods, publish and subscribe. You will learn how to access your database and server shells using command line tools. You will use the SimpleSchema system to validate data and generate input forms automatically. You will see a complete collaborative code editing environment, TextCircle, being built from scratch.

At the end of this course, you will be able to: - use Meteor methods to control data write access - use publish and subscribe to control data read access - install and use advanced Meteor packages - add user accounts to your applications - implement complex MongoDB filters - use the MongoDB and meteor server shells - define data validations schemas using SimpleSchema - generate data input forms automatically using SimpleSchema In this course, you will complete: 2 programming assignments taking ~4 hours each to complete 4 quizzes, each taking ~20 minutes to complete multiple practice quizzes, each taking ~5 minutes to complete

W E E K 1:

Web Application Development with JavaScript and MongoDB: Course overview

Welcome to 'Web Application Development with JavaScript and MongoDB'! In this course we will be creating native mobile apps using Meteor.js, implementing social media features, such as following as well as writing and running unit tests on your JavaScript code. Finally you will set up your own server environment to run Meteor applications and you will implement the publish and subscribe data control model. I hope you enjoy the course! -Matthew

MongoDB, Meteor and reactive data

Welcome to the first module of 'Web Application Development with JavaScript and MongoDB!' In this module we will look at accessing MongoDB on the command line, we will understand how to check for valid returns from find queries and identify reactive data sources within the Meteor framework. Finally we learn how to use the Session object to store user data, understand variable scope and use iframes to create separate DOMs. Enjoy!

W E E K 2

User accounts, packages and methods

Welcome to the second module of 'Web Application Development with JavaScript and MongoDB!' In this module we will learn how to use the core user accounts packages, customise the user accounts UI using third party packages and search for and add packages to an application. We will also look at how to query MongoDB collections from the command line and learn how to control data write access using methods. Enjoy!

W E E K 3

Publish and subscribe model and MongoDB filters

Welcome to the third module of 'Web Application Development with JavaScript and MongoDB!' In this module we will use bootstrap icons and use the publish and subscribe model to control data read access. We will understand the concept of asynchronous execution and work with template data contexts and helper functions. Finally we will use packages to implement in-place content editing and use complex MongoDB filters. Enjoy!

W E E K 4

SimpleSchemas, autoform and code re-organisation

Welcome to the final module of 'Web Application Development with JavaScript and MongoDB!' In this module we will create a well organised application using special Meteor folders and we will organise templates into multiple files. We will also learn how to use the iron:router package to create multiple page applications and we will generate data entry forms automatically using SimpleSchema and autoform. Finally, we will validate user data automatically using SimpleSchema.

Semester VII
Functional Programming Paradigms
(Mentor: Dr. Aditya Desai)

1. **Cloud computing**

Link: https://swayam.gov.in/nd1_noc20_cs65/preview (NPTEL)

C O U R S E L A Y O U T

Week 1: Introduction to Cloud Computing
Week 2: Cloud Computing Architecture
Week 3: Service Management in Cloud Computing
Week 4: Data Management in Cloud Computing
Week 5: Resource Management in Cloud
Week 6: Cloud Security
Week 7: Open Source and Commercial Clouds, Cloud Simulator
Week 8: Research trend in Cloud Computing, Fog Computing

B O O K S A N D R E F E R E N C E S

1. Cloud Computing: Principles and Paradigms, Editors: Rajkumar Buyya, James Broberg, Andrzej M.

Goscinski, Wiley,2011

2. Enterprise Cloud Computing - Technology, Architecture, Applications, Gautam Shroff, Cambridge University Press, 2010
3. Cloud Computing Bible, Barrie Sosinsky, Wiley-India, 2010
4. Cloud Security: A Comprehensive Guide to Secure Cloud Computing, Ronald L. Krutz, Russell Dean Vines, Wiley- India,2010

2. Functional Programming Principles in Scala

Link: https://www.coursera.org/learn/progfun1?specialization=scala

Functional programming is becoming increasingly widespread in industry. This trend is driven by the adoption of Scala as the main programming language for many applications. Scala fuses functional and object-oriented programming in a practical package. It interoperates seamlessly with both Java and Javascript. Scala is the implementation language of many important frameworks, including Apache Spark, Kafka, and Akka. It provides the core infrastructure for sites such as Twitter, Tumblr and also Coursera.

In this course you will discover the elements of the functional programming style and learn how to apply them usefully in your daily programming tasks. You will also develop a solid foundation for reasoning about functional programs, by touching upon proofs of invariants and the tracing of execution symbolically. The course is hands on; most units introduce short programs that serve as illustrations of important concepts and invite you to play with them, modifying and improving them. The course is complemented by a series programming projects as homework assignments. Recommended background: You should have at least one year programming experience. Proficiency with Java or C# is ideal, but experience with other languages such as C/C++, Python, Javascript or Ruby is also sufficient. You should have some familiarity using the command line

W E E K 1:

Getting Started + Functions & Evaluation

Get up and running with Scala on your computer. Complete an example assignment to familiarize yourself with our unique way of submitting assignments. In this week, we'll learn the difference between functional imperative programming. We step through the basics of Scala; covering expressions, evaluation, conditionals, functions, and recursion

W E E K 2:

Higher Order Functions

This week, we'll learn about functions as first-class values, and higher order functions. We'll also learn about Scala's syntax and how it's formally defined. Finally, we'll learn about methods, classes, and data abstraction through the design of a data structure for rational numbers.

W E E K 3:

Data and Abstraction

This week, we'll cover traits, and we'll learn how to organize classes into hierarchies. We'll cover the hierarchy of standard Scala types, and see how to organize classes and traits into packages. Finally, we'll touch upon the different sorts of polymorphism in Scala.

W E E K 4:

Types and Pattern Matching

This week we'll learn about the relationship between functions and objects in Scala; functions *are* objects! We'll zoom in on Scala's type system, covering subtyping and generics, and moving on to more advanced aspects of Scala's type system like variance. Finally, we'll cover Scala's most widely used data structure, Lists, and one of Scala's most powerful tools, pattern matching.

3. **Parallel programming**
   Link: https://www.coursera.org/learn/parprog1?specialization=scala

With every smartphone and computer now boasting multiple processors, the use of functional ideas to facilitate parallel programming is becoming increasingly widespread. In this course, you'll learn the fundamentals of parallel programming, from task parallelism to data parallelism. In particular, you'll see how many familiar ideas from functional programming map perfectly to to the data parallel paradigm. We'll start the nuts and bolts how to effectively parallelize familiar collections operations, and we'll build up to parallel collections, a production-ready data parallel collections library available in the Scala standard library. Throughout, we'll apply these concepts through several hands-on examples that analyze real-world data, such as popular algorithms like k-means clustering.

Learning Outcomes. By the end of this course you will be able to: - reason about task and data parallel programs, - express common algorithms in a functional style and solve them in parallel, - competently microbenchmark parallel code, - write programs that effectively use parallel collections to achieve performance Recommended background: You should have at least one year programming experience. Proficiency with Java or C# is ideal, but experience with other languages such as C/C++, Python, Javascript or Ruby is also sufficient. You should have some familiarity using the command line. This course is intended to be taken after Functional Program Design in Scala: https://www.coursera.org/learn/progfun2.

W E E K 1:

Parallel Programming

We motivate parallel programming and introduce the basic constructs for building parallel programs on JVM and Scala. Examples such as array norm and Monte Carlo computations illustrate these concepts. We show how to estimate work and depth of parallel programs as well as how to benchmark the implementations.

W E E K 2:

Basic Task Parallel Algorithms

We continue with examples of parallel algorithms by presenting a parallel merge sort. We then explain how operations such as map, reduce, and scan can be computed in parallel. We present associativity as the key condition enabling parallel implementation of reduce and scan.

W E E K 3:

Data-Parallelism

We show how data parallel operations enable the development of elegant data-parallel code in Scala. We give an overview of the parallel collections hierarchy, including the traits of splitters and combiners that complement iterators and builders from the sequential case.

W E E K 4:

Data Structures for Parallel Computing

We give a glimpse of the internals of data structures for parallel computing, which helps us understand what is happening under the hood of parallel collections.

4. Functional Programming Scala in Capstone
Link: https://www.coursera.org/learn/scala-capstone

In the final capstone project you will apply the skills you learned by building a large data-intensive application using real-world data.

You will implement a complete application processing several gigabytes of data. This application will show interactive visualizations of the evolution of temperatures over time all over the world. The development of such an application will involve: — transforming data provided by weather stations into meaningful information like, for instance, the average temperature of each point of the globe over the last ten years ; — then, making images from this information by using spatial and linear interpolation techniques ; — finally, implementing how the user interface will react to users' actions.

Syllabus - What you will learn from this course

W E E K 1
Project overview

Get an overview of the project and all the information to get started. Transform data provided by weather stations into meaningful information.

W E E K 2
Raw data display

Transform temperature data into images, using various interpolation techniques.

W E E K 3
Interactive visualization

Generate images compatible with most Web-based mapping libraries.

W E E K 4
Data manipulation

Get more meaning from your data: compute temperature deviations compared to normals.

Semester VIII
Haskell Programming
(Mentor: Mr. Sandeep Bankar)

1. **Introduction To Haskell Programming**

Link: https://swayam.gov.in/nd1_noc20_cs79/preview

COURSE LAYOUT

Week 1:	Introduction to Haskell and the ghci interpreter
Week 2:	Defining functions: guards, pattern matching and recursion
Week 3:	Lists, strings and tuples
Week 4:	Types and polymorphim
Week 5:	Higher order functions on lists: map, filter, list comprehension
Week 6:	Computation as rewriting, lazy evaluation and infinite data structures
Week 7:	Conditional polymorphism and type classes
Week 8:	User defined datatypes: lists, queues, trees
Week 9:	Input/output and the ghc compiler
Week 10:	Arrays