PUNYASHLOK AHILYADEVI HOLKAR
SOLAPUR UNIVERSITY, SOLAPUR

NAAC Accredited-2015
'B' Grade (CGPA 2.62)

Name of the Faculty: Science and Technology

Syllabus: B. Sc. (Computer Science) Part-III

Name of the Programme: B. Sc. (CS) Part-III (Sem–V and VI)

(Syllabus to be implemented w.e.f. June 2024)

Punyashlok Ahilyadevi Holkar Solapur University, Solapur

B.Sc. (Computer Science)

Preamble:

Welcome to the Bachelor of Science program in Computer Science, where curiosity meets computation and innovation knows no bounds. In this advanced stage of your academic journey, we delve deeper into the intricate complexities of the digital realm, exploring cutting-edge technologies and emerging trends.

As you embark on this next chapter of your education, prepare to be challenged, inspired, and empowered to push the boundaries of what is possible in the field of computer science. Our esteemed faculty members, experts in their respective domains, are committed to guiding you on this transformative path towards excellence.

Throughout this program, you will have the opportunity to engage in rigorous coursework, hands-on projects, and research endeavors that will not only deepen your understanding of computer science but also sharpen your critical thinking, problem-solving, and communication skills.

At the heart of our curriculum lies a commitment to fostering innovation and creativity. Whether you aspire to develop groundbreaking algorithms, design revolutionary software, or harness the power of artificial intelligence, our program provides the tools, resources, and mentorship to turn your dreams into reality.

Beyond the classroom, you will have access to state-of-the-art facilities, industry partnerships, and internship opportunities that will allow you to gain real-world experience and make meaningful contributions to the field.

As you navigate through this program, remember that the journey of learning is a continuous process—one that requires dedication, perseverance, and a thirst for knowledge. Embrace every challenge as an opportunity for growth, and every setback as a chance to learn and improve.

Together, let us embark on this exhilarating adventure, united in our passion for innovation, discovery, and the pursuit of excellence in computer science. We are excited to embark on this educational journey with you and to support your growth and success in the field of computer science. Welcome to the Bachelor of Science program in Computer Science where the future of technology awaits, and the possibilities are endless.

Programme Objectives:

1. Equip students with a solid understanding of fundamental principles and theories in computer science.
2. Develop proficiency in programming languages and software development methodologies.
3. Foster analytical and problem-solving skills essential for tackling complex computational challenges.
4. Equip students with professional skills pertinent to the Software Industry.
5. Cultivate adaptability to emerging technologies and innovations in the field of Computer Science.
6. Prepare students for advanced study or careers in diverse areas such as software engineering, data science, artificial intelligence, and cybersecurity.
7. Cultivate pathways for successful careers in Computer Science and foster entrepreneurial spirit for software innovation.
8. Foster a deep understanding of theoretical concepts along with their practical applications across various domains.
9. Encourage critical thinking and creativity to explore novel solutions in computer science.
10. Instill ethical values and professional integrity in the practice of computing.

Programme Outcomes (PO):

These outcomes delineate the knowledge and skills students are anticipated to possess upon graduation. They encompass the competencies, knowledge, and behaviors cultivated throughout the program, reflecting the culmination of their educational journey.

Programme Outcomes for B.Sc. in Computer Science:

1. Demonstrate a strong understanding of fundamental principles and theories in computer science, including algorithms, data structures, and computer architecture.
2. Possess the ability to develop software solutions using various programming languages, tools, and methodologies.
3. Apply analytical and critical thinking skills to identify, analyze, and solve complex computational problems.
4. Adapt to emerging technologies and trends in the computing field, and effectively utilize them in practical applications.
5. Communicate technical concepts and solutions clearly and effectively, both orally and in writing, to diverse audiences.

6. Collaborate effectively in multidisciplinary teams to accomplish common goals and projects in computer science and related fields.
7. Engage in lifelong learning and professional development to stay abreast of advancements in technology and maintain relevance in the computing industry.
8. : Understand and adhere to ethical standards and professional responsibilities in the practice of computer science, including issues related to privacy, security, and intellectual property.
9. Design and implement software solutions to address real-world challenges, meeting the needs of both the computing field and society.
10. Establish connections between computer science and other auxiliary disciplines, enhancing interdisciplinary problem-solving capabilities.
11. Demonstrate ethical conduct in utilizing internet and cyber systems, adhering to established codes of behavior.
12. Prepare for advanced studies in specialized areas and pursue technical careers in the industry.
13. Manage, deploy, and configure computer networks, hardware, and software operations within organizational settings.
14. Design and create computer programs and systems related to algorithms, networking, web design, cloud computing, IoT, and data analytics.
15. Stay updated with current trends in industrial and research environments, fostering innovation for addressing existing challenges.

Punyashlok Ahilyadevi Holkar Solapur University, Solapur

Faculty of Science & Technology

Choice Based Credit System (CBCS)(w.e.f.2024-25)

Revised Structure for B. Sc-III

Subject/ Core Course	Name and Type of the Paper		No. of papers/ Practical	Hrs/week			Total Marks Per Paper	UA	CA	Credits
	Type	Name		L	T	P				
Class :	**B.Sc.- III Semester - V**									
Ability Enhancement Course(AECC)		English (Business English)	Paper II Part A	4	--	--	50	40	10	2.0
Core Courses: (Students can opt any one subjects among the three Subjects excluding Interdisciplinary/Additional subject offered at B. Sc-II.)	DSC 1 E	Data Communication and Networking	Paper IX	4	--	--	100	80	20	4.0
	DSC 1 F	Advanced Java	Paper X	4	--	--	100	80	20	4.0
	DSC 1 G	Dot Net Core	Paper XI	4	--	--	100	80	20	4.0
	DSE 1 A	Python Programming	Paper XII	4	--	--	100	80	20	4.0
	DSE 1 B	Kotlin Programming								
	DSE 1 C	Data warehousing and Mining								
	(Select either A or B or C from DSE 1)									
Total Theory Sem-V				**20**	**--**	**--**	**450**	**360**	**90**	**18**
	$ SEC-2	Software Testing		4	--	--	100	80	20	4.0
Class :	**B.Sc.- III Semester –VI**									
Ability Enhancement Course(AECC)		English (Business English)	Paper II Part B	4	--	--	50	40	10	2.0
Core Courses: (Students can opt any one subjects among the three Subjects excluding interdisciplinary / Additional subject offered at B.Sc. II.	DSC 1 H	Network Security	Paper XIII	4	--	--	100	80	20	4.0
	DSC 1 I	Linux and Shell Programming	Paper XIV	4	--	--	100	80	20	4.0
	DSC 1 J	ASP.Net Core MVC	Paper XV	4	--	--	100	80	20	4.0
	DSE 2 A	Advanced Python	Paper XVI	4	--	--	100	80	20	4.0
	DSE 2 B	Mobile Application Development using Kotlin								
	DSE 2 C	Data Visualization using Power BI								
	(Select either A or B or C from DSE 2)									
Total Theory Sem-VI				**20**	**-**	**--**	**450**	**360**	**90**	**18**
Core		Project	Practical IV	--	-	5	100	80	20	4.0
		DSC 1F & 1 I	Practical V	--	-	5	100	80	20	4.0
		DSC 1G & 1 J	Practical VI	--	-	5	100	80	20	4.0
		DSE 1 A/B/C & 2 A/B/C	Practical VII	--	-	5	100	80	20	4.0
Total (Practical)						**20**	**400**	**320**	**80**	**16**
Grand Total				**40**		**20**	**1300**	**1040**	**260**	**52**
	$ SEC- 2			**4**			**100**	**80**	**20**	**4**

$The students can choose MOOCs/ NPTEL/SWAYAM/Pathshala/Add-on / Skill based courses of university/college initiated courses of same credits.

$ These courses are not compulsory, but after completion of these courses students get additional credits on their Mark lists. $SEC Courses initiated by colleges should be communicated to university for information and necessary action.

Equivalence papers for B. Sc.- III Sem V and VI (Computer Science)

Sr. No	Old Paper		New Paper (Equivalence papers)	
	Paper Type and Number	**Paper Name**	**Paper Type and Number**	**Paper Name**
1	DSE 1 A Paper IX	Visual Programming Using C#	DSC 1 G Paper XI	Dot Net Core
2	DSE 2 A Paper X	Core Java	No Equivalence	
3	DSE 3 A Paper XI	Operating System	No Equivalence	
4	DSE 4 A Paper XII	Python	DSE 1 A Paper XII	Python Programming
5	SEC 3 Paper XIII	Linux	DSC 1 I Paper XIV	Linux and Shell Programming
6	DSE 1 B Paper XIV	Web Technology	No Equivalence	
7	DSE 2 B Paper XV	Advanced Java	DSC 1 F Paper X	Advanced Java
8	DSE 3 B Paper XVI	Data Communication and Networking	DSC 1 E Paper IX	Data Communication and Networking
9	DSE 4 B Paper XVII	Advance Python	DSE 2 A Paper XVI	Advanced Python
10	SEC 4 Paper XVIII	Software Testing	$ SEC-2	Software Testing

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V						
Paper Type:	DSC 1 E			**Paper No:**	Paper IX		
Paper Name:	Data Communication and Networking						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. Study the basic taxonomy and terminology of the computer networking and enumerate the layers of OSI model and TCP/IP model.
2. Acquire knowledge of Application layer and Presentation layer paradigms and protocols.
3. Study Session layer design issues, Transport layer services, and protocols.
4. Study data link layer concepts, design issues, and protocols.
5. Read the fundamentals and basics of Physical layer, and will apply them in real time applications.

Unit 1: - Introduction to Data Communication & Networking (10)

Data Communication: Components, Data Flow, Protocols & Standards, Design Issues of Layers, Connection oriented and connection less services, Network models :- ISO-OSI reference model, TCP/IP reference model.

Unit 2:- Physical layer (20)

Signals: Analog & Digital Signals, Period, Frequency, Phase, Amplitude, Bandwidth, Bit Rate, Bit, Length, Fourier analysis. Transmission Impairment: Attenuation, Distortion, Noise, Nyquiest Theorem, Shannon Capacity Theorem. , Transmission Media:-Guided Media-Magnetic Media, Twisted Pair, Coaxial Cable, Fiber Optic Cable, Unguided Media:- Wireless- Radio Waves, Microwaves, Infrared, Satellite Communication, Digital Transmission: Manchester & Differential Manchester Coding, Pulse Code Modulation., Modulation:- Amplitude Modulation, Frequency Modulation, Phase Modulation,. Transmission Mode: Parallel, Serial, Synchronous Transmission, Asynchronous Transmission., Multiplexing- Frequency Division Multiplexing, Time Division Multiplexing, Wavelength Division Multiplexing., Switching- Circuit Switching, Message Switching, Packet Switching.

Unit 3: - Data link layer (15)

Error Detection & Correction: Types of Errors, Hamming Distance, Error Detection: Parity Check, Cyclic Redundancy Check, Checksum Check, hamming code., Data Link Control: Framing, Flow & Error Control, Protocols: Simplex, Stop and Wait, Stop and Wait ARQ, Go Back N ARQ, Selective repeat ARQ, HDLC, Point to Point protocol. Multiple Access Protocol: ALOHA, CSMA, CSMA/CD, CSMA/CA Channelization, FDMA, TDMA, CDMA

Unit 4 :- Network layer , Transport, Session, Presentation & Application layers (15)

Network layer Design issues, Routing Algorithm: Optimality Principle, Shortest Path Routing, Distance Vector Routing, Link State Routing., Congestion Control Algorithm: General principle of congestion control, Congestion, prevention policies, Congestion Control in Virtual-Circuit Subnets, Congestion Control in Datagram Subnets, Network Devices- Hubs, Switches, Repeaters, Bridges, Routers, Gateways, Transport, Session, Presentation & Application layers. TCP/IP protocol suite :- UDP,TCP,SCTP, IP, RTP, FTP, DNS, TELNET, SMTP, POP, HTTP, WWW, SNMP,ARP, RARP., Data Compression:-Audio Compression, Video Compression

Course Outcomes: -

Students will able to:

1. Describe the functions of each layer in OSI and TCP/IP model.
2. Explain the functions of Application layer and Presentation layer paradigms and Protocols.
3. Describe the Session layer and Transport layer.
4. Describe the functions of data link layer and explain the protocols.
5. Explain the types of transmission media with real time applications

Reference Books:

1. Computer Networking by Tannenbaum.
2. Data communication and networking by William Stallings
3. Data communication and networking by B A Forouzan
4. Data communication and networking by Jain

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V						
Paper Type:	DSC 1 F			**Paper No:**	Paper X		
Paper Name:	Advanced Java						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. To understand database connectivity using JDBC.
2. To learn how to develop web applications using servlet.
3. How to develop web applications using JSP.
4. To Understand concept of hibernate and struts.

Unit -1: -JDBC **(10)**

Introducing JDBC: Describing Components of JDBC, Features of JDBC, JDBC Architecture: Types of Divers: Advantages and disadvantages of Drives, Use of Drivers, JDBC Statement and Methods:-Statement, PreparedStatement, CallableStatement, execute(), executeQuery(), executeUpdate(), Working with Resultset interface, Working with Resultset and MetaData.

Unit -2: -Servlet **(15)**

Introducing CGI, Introducing Servlet, Advantages of Servlet over CGI, Features of Servlet, Introducing Servlet API, Javax.servlet package, Javax.servlet.http package, Introducing Servlet, Advantages of Servlet over CGI, Features of Servlet, Servlet life Cycle, Init(), Service(), Destroy(), Working with GenericServlet and HttpServlet, RequestDispatcher interface, Include() and forward(), Use of RequestDispatcher, Session in Servlet, Introducing session, Session tracking mechanism, Cookies, Advantages & disadvantages, use of cookies, Hidden form filed, Advantages & disadvantages, use of Hidden form filed, URL rewritten, disadvantages, use of URL rewritten, HttpSession, Advantages & disadvantages,use of URL HttpSession

Unit -3: - JSP **(13)**

Introduction to JSP, Advantages of JSP over Servlet, JSP architecture, JSP life cycle, Implicit objects in JSP- request, response, out, page, pageContext, application, session, config, exception, JSP tag elements- Declarative, Declaration, scriplet, expression, action., Java Bean- Advantages & Disadvantages, useBean tag- setProperty and getProperty, Bean In Jsp, JSTL core tag: General purpose tag, conditional tag, networking tag, JSTL SQL tags, Custom tag: empty tag, body content tag, iteration tag, simple tag

Unit -4: - Hibernate and Spring **(20)**

Introduction to Hibernate (HB), Architecture of HB, Generator classes, Steps to create application of HB, HB with annotation, Insert , Delete, update, retrieve records from database in HB, HB web Application **Spring**-Introduction to spring, Spring modules, Spring application, Spring Framework, Dependency injection, Annotation, constructor Injection (CI), CI dependant object, CI with collection, CI with map, CI inheriting bean, Spring JDBC: JDBC template, PreparedStatement, ResultsetExactor, RowMapper, NamedParameter, Simple JDBC template, Spring with Hibernate, Spring MVC, Spring ORM Theory, Spring Data JPA

Course Outcomes: -

Students will be able to:

1. Use database connectivity using JDBC.
2. Develop web applications using servlet.
3. Develop web applications using JSP.
4. To use the concept of hibernate and struts.

Reference Books

1. Java The complete Reference by Herbert Schildt
2. Java Servlet Programming by Jasan Hunter
3. Beginning Java EE5 from Novice to Professionals by K. Makhar & C. Zelenk
4. Java Server Programming by Bayross & Shah
5. Thinking in java by Brucel
6. Spring Persistence with Hibernate- Paul Tepper Fisher, Brian D Murphy
7. https://docs.spring.io/spring-framework/docs/4.3.25.RELEASE/spring-framework-reference/pdf/spring-framework-reference.pdf

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V						
Paper Type:	DSC 1 G			**Paper No:**	Paper XI		
Paper Name:	Dot Net Core						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. To understand how to design, implement, test, debug, and document programs that use basic data types and computation, simple I/O, conditional and control structures, string handling and functions in C#.
2. To understand the importance of Classes & objects along with constructors, Arrays and Vectors in C#.
3. Discuss the principles of inheritance, interface and demonstrate though problem analysis assignments how they relate to the design of methods, abstract classes and interfaces and packages in C#.
5. To understand importance of Multi-threading & different exception handling mechanisms in C#.
6. To understand basic idea about how to design GUI base windows application using C#.

Unit 1: - Introduction to C# (10)

Understanding .NET- The .NET Framework, .NET Core, Download and install C# Development Environments - Visual Studio, Visual Studio Code, building console apps using Visual Studio 2022 and Building console apps using Visual Studio Code, C# Basics- Variables & Data Types, Reference & Value Types-Nullable types, Elvis operator, Null coalescing operator, Boxing & unboxing, Keywords, Initialization, Type Inference, Console Input & Output., Operators, Operator precedence, Type conversion, C# statements- Branching, Jumping, Looping, Complex data types- Enums, Arrays, Tuples

Unit 2: - Object Oriented Programming (15)

Classes and object-Declaration, Access modifiers, Data, Methods, Method parameters, Constructors, Deconstruct, Method overloading, Properties, Local and global variable and methods, Static classes, methods & members, nested classes, Indexers, Partial types & methods, Structs & Records, Inheritance- Base & derived classes, advantages, Types. Constructors in inheritance. Abstract classes, sealed class, Interfaces - Defining & implementing, Default interface methods, Interface inheritance, .NET interfaces, Polymorphism- Virtual methods. Method overriding, operator overloading, Abstract methods, Sealed types,

,

Unit 3: - Threading, exception and resource management (15)

Exception- about exception, Exceptions Hierarchy, Throwing and Catching Exception, The try-finally Construct, IDisposable and the "using" Statement, Advantages of Using Exceptions, inbuild exception, custom exception, Threading-about threading, Thread Name, Thread Priority, and Thread State, Foreground and background threads in C#, Multithreading - An Overview, The Thread Class, ThreadPool Threads, Collections- Generic collections, Concurrent collections, Specialized collections, Performance considerations, Resource Management- Finalizers, Garbage Collection, IDisposable, The using statement, Serialization-Attributes, JSON serialization, Binary serialization, XML serialization,

Unit 4: - Delegate, event and LINQ (20)

Delegates-Multicast delegates, generic delegates, Action<T>, Predicate<T>, Func<T> , Lambdas-Expression & statement lambdas, Parameters, Return type, Captures, Events- Defining, Raising, Standard & custom events, LINQ-Enabling features, LINQ expression, LINQ pattern, Joins, Aggregations, Basic of Windows application

Course Outcomes: -

Students will be able to:

1. Gain proficiency in designing, implementing, testing, debugging, and documenting programs utilizing fundamental data types, computation, basic I/O, conditional and control structures, string manipulation, and functions within the C# programming language.
2. Appreciate the significance of classes, objects, constructors, arrays, and vectors within the C# framework.
3. Explore the principles of inheritance and interfaces, demonstrating their application through problem analysis assignments and their relevance to method design, abstract classes, interfaces, and packages in C#.
4. Recognize the importance of multi-threading and various exception handling mechanisms in C# programming.
5. Acquire a foundational understanding of designing GUI-based Windows applications using C#.

Reference books:

1. C# 12 and .NET 8 - Modern Cross-Platform Development Fundamentals - Eighth Edition: Start building websites and services with ASP.NET Core 8, Blazor, and EF Core 8 8th ed. Edition by Mark J Price, Packt Publishing; 8th ed. edition (November 14, 2023)
2. Professional C# and .NET - 2021 Edition by by Christian Nagel, Wrox Press book
3. C# 12 in a Nutshell: The Definitive Reference 1st Edition by Joseph Albahari, O'Reilly Media;
4. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming by by Andrew Troelsen & Phil Japikse, Apress

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V						
Paper Type:	DSE 1 **A (Elective)**			**Paper No:**	Paper XII		
Paper Name:	Python Programming						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. Basics of Python programming
2. Decision Making and Functions in Python
3. Object Oriented Programming using Python
4. Files Handling in Python
5. Regular expression for pattern matching

Unit 1: - Introduction to Python: (10)

Features/Characteristics of Python, Installation and Working with Python, Structure of a Python Program, Writing simple python program, Executing python program using command line window and IDLE graphics window, Python Virtual Machine, Identifiers and Keywords, Operators (Arithmetic operators, Relational operators, Logical or Boolean operators, Assignment Operators, Bit wise operators, Membership operators, Identity operators), Operator Precedence and Associativity, Python Data Types: -Python Variables, mutable and immutable types Data types in python, Built-in Datatypes, Sequences in python, Literals in python, User Defined Datatypes, Constants in python, Type conversion, Input and Output Statements, Command line arguments, Control Statements:-Conditional Statements: if, if-else, nested if –else, Looping: for, while, nested loops, Loop manipulation using pass, continue, break, assert and else suite

Unit 2:- Strings, Collection Lists, Tuples, Dictionaries, Functions and, Modules (15)

Strings: Introduction to String, String Manipulation., Collection List: Introduction to List, manipulating list., Tuples: Introduction to Tuples, Manipulating Tuples., Dictionaries: Concept of Dictionary, Techniques to create, update & delete dictionary items., Functions, Modules :- Difference between a Function and a Method, Functions:- Defining a function, Calling a function, Advantages of functions, Types of functions, Function parameters:-types of parameter-default, keyword base, variable length, Formal parameters, Actual parameters, Anonymous functions, Global and Local variables, Modules:- Importing module, Creating & exploring modules, Math module, Random module, Date and Time module, RE module

Unit 3: - Object Oriented Programming (20)

Features, Concept of Class & Objects, Constructor, Types of Variables, Namespaces, Types of Methods, Inner Classes, Constructors in Inheritance, Overriding Super Class Constructors and Methods, Types of Inheritance, Abstract Classes and Interfaces, The Super() Method, Operator Overloading, Method Overloading, Method Overriding, MRO

Unit 4: Threading, Exception Handling and File (15)

Threading- about Thread, Starting a Thread, Daemon Threads, join() a Thread, Working With Many Threads, Thread, Race Conditions, Synchronization, Deadlock, Exception :- Errors in a Program, Exceptions, Exception handling, Types of Exceptions, User defined Exceptions, Python File Operation:- Types of File, Opening and Closing a File, Reading and writing to files, Manipulating directories and files.

Course Outcomes: -

Students will be able to:

1. Describe the Numbers, Math functions, Strings, List, Tuples and Dictionaries in Python
2. Express different Decision Making statements and Functions
3. Interpret Object oriented programming in Python
4. Understand and summarize different File handling operations
5. Understand Regular expression and implement for pattern matching.

Reference Books

1. Beginning Python by Magnus Lie Hetland-Apress
2. Python Programming for the Absolute Beginner by Michael Dawson-Cengage Learning
3. Python for Everybody: Exploring Data in Python 3 by Charles Severance-CreateSpace
1. Independent Publishing Platform
4. Introducing Python: Modern Computing in Simple Packages by Bill Lubanovic-O'Reilly
2. Media
5. Python Programming for Beginners: An Introduction to the Python Computer by Jason
3. Cannon- CreateSpace Independent Publishing Platform
6. Python for Beginners by Harsh Bhasin

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V						
Paper Type:	DSE 1 **B (Elective)**			**Paper No:**	Paper XII		
Paper Name:	Kotlin Programming						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. The Kotlin programming language.
2. Familiarize with Kotlin syntax, features, and basic programming concepts.
3. Explore language features such as type inference, null safety, and extension functions, and understand their importance in Kotlin programming.
4. Provide a solid foundation in object-oriented programming (OOP) concepts within the context of Kotlin which covers classes, objects, inheritance, interfaces, and other OOP principles, and demonstrate their implementation in Kotlin.
5. To introduce functional programming concepts in Kotlin, including higher-order functions, lambda expressions, and functional manipulation of collections and understand how functional programming enhances code readability, maintainability, and reusability.
6. Equip the skills to write Kotlin code for simple applications.
7. Lay the foundation for further exploration and application of Kotlin in software development.

Unit 1: - Introduction to Kotlin **(15)**

Introduction to Kotlin, Download and Install JDK, Download and Install IntelliJ Idea, Creating New Kotlin Project, Creating Kotlin File, main Function, Run Kotlin Application, Single Line and MultiLine Comment, Data Type- basic data types (Int, String, Boolean, etc.), Variable using var and val, type inference, Operators and expressions: Arithmetical Operators, Comparison Operators ,Equality and Inequality Operators ,Logical Operators ,Increment and Decrement Operators ,Augmented Assignments etc., Control flow: if Expression, if else Expression ,when Expression, for Loop ,while Loop ,do while Loop ,Break and Continue, Getting User Input using readLine, Getting User Input using Scanner, String ,String Concatenation ,String Literals - Escaped String ,String Literals - Raw String ,String Templates

Unit 2: - Functions and Lambdas **(15)**

Defining functions: function syntax, parameters, return types., Function without Parameters, Function with Parameters, Function with Default Argument, Function with Named Argument, Higher-order functions: passing functions as arguments, returning functions from functions. Lambda expressions: syntax, usage, capturing variables., Anonymous Function, Null Safety- Nullable and non-nullable types in Kotlin., Safe calls (?.), Elvis operator (?:), and the !! operator., Handling nullability: safe casts (as?), let, run, apply, also functions

Unit 3: - Collections **(10)**

Introduction to Kotlin collections: array lists, sets, maps or Dictionary, Basic operations on collections: iterating, adding, removing elements., Functional programming with collections: map, filter, reduce.

Unit 4: - Object-Oriented Programming **(20)**

Classes and objects: class declaration, properties, methods., Constructors: primary constructors, secondary constructors., Getter and Setter, Inheritance: superclass, subclass, Inheritance with Constructor, overriding methods., Properties and Function, Super, Visibility Modifiers, Abstract Class and Method, Interfaces: defining interfaces, implementing interfaces in classes., Data Class, Object Destructuring, Exception Handling: try-catch block, throw keyword, finally block., Kotlin's approach to checked and unchecked exceptions., File I/O:-Reading from and writing to files., Working with text files: reading lines, writing to files., Handling file exceptions.

Course Outcomes: -

Students will be able to:

1. Gain a comprehensive understanding of the Kotlin programming language.
2. Acquire familiarity with Kotlin syntax, features, and fundamental programming concepts.
3. Explore and grasp the significance of language features such as type inference, null safety, and extension functions in Kotlin programming.
4. Develop a strong foundation in object-oriented programming (OOP) principles within the Kotlin context, covering classes, objects, inheritance, interfaces, and other related concepts.
5. Introduce functional programming concepts in Kotlin, including higher-order functions, lambda expressions, and functional manipulation of collections, while understanding their impact on code readability, maintainability, and reusability.
6. Develop the skills necessary to write Kotlin code for simple applications.
7. Establish a groundwork for further exploration and utilization of Kotlin in software development endeavors.

References:

1. "Kotlin Programming: The Big Nerd Ranch Guide" by Josh Skeen and David Greenhalgh
2. "Kotlin in Action" by Dmitry Jemerov and Svetlana Isakova
3. Online Kotlin documentation and tutorials.

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V						
Paper Type:	DSE 1 **C (Elective)**			**Paper No:**	Paper XII		
Paper Name:	Data warehousing and Mining						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. To gain a solid understanding of the principles, concepts, and architecture of data warehousing systems, including data storage, extraction, transformation, and loading (ETL) processes.
2. Learn various data modeling techniques such as star schema, snowflake schema, and dimensional modeling to design efficient and scalable data warehouse schemas.
3. To gain expertise in association rule mining technique.
4. To develop proficiency in implementing and interpreting various data mining classification and prediction algorithms from large datasets.
5. To develop proficiency in implementing and interpreting various data mining clustering algorithms from large datasets.
6. To develop skills for using recent data mining software to solve practical problems in a variety of disciplines.
7. Apply data mining techniques to real-world datasets and scenarios, including customer segmentation, market basket analysis, fraud detection, and predictive analytics.

Unit 1: - Introduction to Data Warehouse and Data mining (10)

Introduction: What is Data Warehouse? Differences between Operational Database Systems and Data Warehouses, Data Warehouse Architecture, Data Warehouse Components, A Multidimensional Data Model, Schemas, Data Warehouse Implementation, Data cube Technology, OLAP operations, Data mining query language, Data Mining:- What is data mining, Evolution, KDD, What kind of data, Architecture, data mining views, Data Mining Functionalities, Issues in Data Mining

Unit 2: - Data Preprocessing and Association Rule mining (10)

Data Preprocessing:- An Overview, Extract, Transform, Load (ETL) Processes, Data Cleaning, Data Integration, Data Transformation and Data Discretization, Data Reduction., Frequent Patterns, Associations, and Correlations:- Market Basket Analysis, Frequent Itemsets, Closed Itemsets, and Association Rules, Frequent Itemset Mining Methods-Apriori Algorithm: Finding Frequent Itemsets, Generating Association Rules from Frequent Itemsets, Improving the Efficiency

of Apriori, A Pattern-Growth Approach for Mining Frequent Itemsets, Mining Multilevel and multidimensional Association Rules, Constraint-Based Frequent Pattern Mining

Unit 3: - Supervised Learning Technique (20)

supervised and unsupervised learning, What Is Classification? What is regression, difference between classification and regressing, General Approach to Classification, Issues regarding Classification and Predication, Binary and Multiclass Classification, Types of classifications, Classification by Decision tree induction, Bayesian Classification, Classification by Back propagation, Logistic regression, k-Nearest-Neighbor Classifiers, SVM, Introducing Ensemble Methods- Bagging, Boosting, AdaBoost, Random Forests, Other classification methods, Prediction: regression. Model Evaluation and Selection-Metrics for Evaluating Classifier Performance, Cross-Validation, underfitting and overfitting

Unit 4: - Unsupervised Learning Technique and Applications (20)

Clustering: What is Cluster Analysis? Types of data in Cluster Analysis, A Categorization of Major Clustering Methods., Partitioning Methods, Hierarchical Methods, Density-Based Methods, Model-Based Clustering Methods: Statistical Approach, Neural Network Approach, Outlier Analysis, Applications and Trends in Data Mining: Data Mining Applications, Data Mining for Financial Data Analysis, Data Mining for Retail and Telecommunication Industries, Data Mining in Science and Engineering, Data Mining for Intrusion Detection and Prevention, Data Mining and Recommender Systems, Spatial Data Mining. Text Data Mining, Multimedia Data Mining, Web Data Mining, Privacy, Security, and Social Impacts of Data Mining, Data Mining and Intelligent Query Answering, Trends in Data Mining.

Course Outcomes: -

Students will be able to:

1. Gain a thorough comprehension of data warehousing systems, encompassing the foundational principles, concepts, and architectural components, including data storage, extraction, transformation, and loading (ETL) processes.
2. Acquire proficiency in employing diverse data modeling techniques such as star schema, snowflake schema, and dimensional modeling to formulate efficient and scalable data warehouse schemas.
3. Developed expertise in association rule mining techniques, enabling the extraction of meaningful patterns and relationships from structured datasets.
4. Developed the ability to proficiently implement and interpret various data mining classification and prediction algorithms, facilitating insightful analysis and decision-making from extensive datasets.
5. Attain proficiency in implementing and interpreting a range of data mining clustering algorithms, empowering the identification of natural groupings and patterns within large datasets.
6. Cultivate skills in utilizing contemporary data mining software to address practical challenges across diverse disciplines, leveraging cutting-edge tools and techniques for effective problem-solving.

7. Apply data mining techniques to real-world datasets and scenarios, including but not limited to customer segmentation, market basket analysis, fraud detection, and predictive analytics, to derive actionable insights and drive informed decision-making.

Reference Books:

1. Data Mining Concepts and Techniques: Jiawei Han and Micheline Kamber, Morgan Kauf Mann Publishers.
2. Modern Data Warehousing, Mining and Visualization: George M. Marakas, Pearson Education,2003.
3. Building the Data Warehouse: W. H. Inmon, Wiley Dreamtech, Third Edition.

B.Sc. (Computer Science)-III (CBCS)							
Sem:	VI						
Paper Type:	DSC 1 H			**Paper No:**	Paper XIII		
Paper Name:	Network Security						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. Introduce students to the fundamental concepts of security, including its necessity, approaches, principles, and various types of security attacks.
2. Provide an overview of cryptography, including plaintext and ciphertext, substitution and transposition techniques, encryption and decryption processes.
3. Introduce symmetric and asymmetric key cryptography algorithms, steganography, key range, key size, and potential types of cryptographic attacks.
4. Investigate Internet security protocols such as Secure Socket Layer/Transport Layer Security (SSL/TLS), Secure Electronic Transaction (SET), and email security protocols like PGP and S/MIME.
5. Cover user authentication basics, including passwords, smart cards, and biometrics, as well as authentication protocols like Kerberos as well as discuss network security principles, firewalls, types of firewalls, and IP security mechanisms to safeguard networks against unauthorized access and malicious activities.

Unit 1: - Fundamental Security Concepts (10)

Security Concepts: Introduction, The need for security, Security approaches, Principles of security, Types of Security attacks – Active and Passive, Security services, Security Mechanisms, A model for Network Security , Access Control Mechanisms: Access Matrix, ACL and capabilities, Access Control Models,.

Unit 2: - Cryptography Concepts and Techniques (20)

Introduction to Cryptography, plain text and cipher text, substitution techniques, transposition techniques, encryption and decryption, symmetric and asymmetric key cryptography, steganography, key range and key size, possible types of attacks, Symmetric Key Cryptographic Algorithms: Algorithm Types and Modes, An overview of Symmetric Key Cryptography, DES, International Data Encryption Algorithm (IDEA), RC5, Blowfish, AES, Asymmetric Key Cryptography: Brief History of Asymmetric Key Cryptography, An overview of Asymmetric Key Cryptography, The RSA Algorithm, Symmetric and Asymmetric Key Cryptography Together

Unit 3: - Digital Signatures and Internet Security Protocols (20)

Digital Signatures: Introduction, Message digests, MD5, SHA-512,MAC, HMAC, Knapsack Algorithm, Elliptic curve Technology, ELGamal Algorithm. , Internet Security Protocols: Secure Socket Layer/TLS, Secure Electronic Transaction, SSL versus SET, E-mail Security- PGP, S/MIME.,

Unit 4: - User Authentication and Network Security (10)

User Authentication and Kerberos: Authentication basics, Passwords, use of smart cards, Biometrics, Kerberos., Network Security: Firewalls, types of firewalls, IP Security, Intrusion : Intruders, Audit Records, Intrusion Detection, honey pots.

Course Outcomes: -

Students will be able to:

1. Familiarize students with the core concepts of security, importance of security measures , including its necessity, approaches, principles, and various types of security attacks.
2. Get an overview of cryptography and encryption and decryption processes.
3. Introduce students to symmetric and asymmetric key cryptography algorithms, steganography, key range, key size, and potential types of cryptographic attacks.
4. Enable students to comprehend the principles and applications of various cryptographic techniques in securing information and communication systems.
5. Examine internet security protocols such as Secure Socket Layer/Transport Layer Security (SSL/TLS), Secure Electronic Transaction (SET), and email security protocols like PGP and S/MIME.
6. Cover the basics of user authentication, including passwords, smart cards, and biometrics, and authentication protocols like Kerberos.
7. Discuss network security principles, firewalls, types of firewalls, and IP security mechanisms to safeguard networks against unauthorized access and malicious activities.

Reference Books:

1. Cryptography and Network Security by Atul Kahate, Tata McGraw-Hill
2. Cryptography and Network Security by Behrouz A. Forouzan, Debdeep Mukhopadhyay, Special Indian Edition, Tata McGraw-Hill.
3. Network Security Essentials: Applications and Standards by William Stallings, Pearson Education.
4. Fundamentals of Computer Security Technology: Edward Amoroso, Prentice-Hall.
5. Cryptography and Data Security: Dorothy E. Denning, Addison-Wesley.
6. Cryptography -Theory and Practice: Douglas R. Stinson, CRC Press.
7. Building Internet Firewalls: D. Brent Chapman and Elizabeth D. Zwicky, O'Reilly andAssociates.

B.Sc. (Computer Science)-III (CBCS)							
Sem:	VI						
Paper Type:	DSC 1 I			**Paper No:**	Paper XIV		
Paper Name:	Linux and Shell Programming						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. To introduce Basic Linux general purpose Commands
2. To learn different editor
3. To learn shell script concepts.
4. To learn file management and permission advance commands.
5. To learn awk, grap, perl scripts.

Unit 1: - Introduction of Linux (10)

History of Linux, Architecture of Linux system & features, Kernel, Shell & its type, Difference between Windows and Linux. Linux Distributions, Working environments: KDE, GNOME, Xface4, Hardware requirement, Installation procedure of Linux, Create partitions, Configuration of X system Users & Groups Management:- Create Users, Create groups, Special groups, Assigning permissions to users and Groups, File and Directory permissions- chmod, chown, chgrp., Linux File System:-Hierarchy of File system, File System parts- Boot Block, Super Block, Inode, Block, Data Block, File types, Devices and Drives in Linux, Mounting devices (CD/DVD, usb, hard drive partition), file system

Unit 2: - Linux Command (15)

Linux commands File and directory Management Commands:-mkdir, rmdir, cd and pwd, file, ls, cat, more, less, File and Directory Operations: find, cp, mv, rm, ln etc, Printing the files - lpr, lpq, lprm etc., Filter Commands & Editor:- Filters: head, tail , pr, cut, paste, sort, uniq, tr, grep, egrep, fgrep, sed., Communication commands:- mesg, talk, write, wall, mail., Text Editors- vi, vim, Archive and File compression commands,

Unit 3: - Linux System Management and Administration (15)

Process Management: Shell process, Parent and children, Process status, System process, Multiple jobs in background and foreground, Changing process priority with nice. Listing processes, ps, kill, premature termination of process., Disk management and System Administration:-Disk Partitioning- RAID, LVM etc., disk related Management Tools- Fdisk, Parted etc. , Boot Loaders- GRUB, LILO, Custom Loaders, System administration – Role of system administrator, identifying administrative tasks & files, Configuration and log files, Chkconfig, Security Enhanced Linux, Installing and

removing packages with rpm command, Understanding various Servers:- DHCP, DNS, Squid, Apache, Telnet, FTP, Samba.

Unit 4: - Shell Programming (20)

Introduction to shell scripting, Writing and executing simple shell scripts, Variables, data types, and operators in shell scripting, Meta characters, Control structure, Loop structure and case statement, Writing and using shell functions, Passing arguments to shell scripts and functions, Returning values from functions, I/O and Redirection, Piping,

Course Outcomes: -

Students will be able to:

1. Identify the basic Linux general purpose commands.
2. Apply and change the ownership and file permissions using advance Linux commands.
3. Use the awk, grep, perl scripts.
4. Implement shell scripts.
5. Apply basic of administrative task.

Reference Books :

1. Official Red Hat Linux Users guide by Redhat, Wiley Dreamtech India
2. Beginning Linux Programming by Neil Mathew & Richard Stones, Wiley Dreamtech India
3. Red Hat Linux Bible by Cristopher Negus, Wiley Dreamtech India
4. UNIX Shell Programming by Yeswant Kanethkar, BPB
5. Shell Scripting: Expert Recipes for Linux, Bash, and More" by Steve Parker
6. Classic Shell Scripting" by Arnold Robbins and Nelson H.F. Beebe
7. Learning the bash Shell: Unix Shell Programming" by Cameron Newham and Bill Rosenblatt

B.Sc. (Computer Science)-III (CBCS)							
Sem:	VI						
Paper Type:	DSC 1 J			**Paper No:**	Paper XV		
Paper Name:	ASP.Net Core MVC						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. Demonstrate the creation of ASP.NET Core MVC Web Applications using .NET 8, covering project file structure, main method, hosting options, and configuration files.
2. Cover Models, Controllers, Views, and Dependency Injection in ASP.NET Core MVC applications.
3. Introduce Entity Framework Core and guide the installation process also explain DbContext in Entity Framework Core and database connection string configuration and database operations.
4. Discuss Transactions, Migration, and Database Seeding in Entity Framework Core.
5. Introduce Partial Views, View Components, and Razor View Engine.
6. Cover Action Results, Routing, Model Binding, HTML Helpers and Tag Helpers in ASP.NET Core MVC.
7. Explain Data Annotations and Model Validations, including custom validations and remote validation.
8. Discuss different methods of State Management, including Cookies and Sessions.

Unit 1: - Introduction to ASP.Net Core MVC **(10)**

Overview of Microsoft Web Technologies, Introduction to ASP.NET Core Framework. NET Core Environment Setup, Download and Install Visual Studio 2022, Download and Install .NET Core SDK, Download and Install SQL Server 2022, Download and Install SSMS, Creating ASP.NET Core Web Application using .NET 8, NET Core Project File Structure, NET Core Main Method, NET Core InProcess Hosting, OutOfProcess Hosting, LaunchSettings.json File, AppSettings.json file, Middleware Components, Web Root (wwwroot) Folder, Static Files Middleware, Configuring Default Page, Developer Exception Page Middleware Command Line Interface, Project Templates in ASP.NET Core Application, Introduction to ASP.NET Core MVC Framework, Set up MVC in ASP.NET Core, Models, Controllers and Views in ASP.NET Core MVC, ASP.NET Core Dependency Injection, Creating ASP.NET Core Application using MVC

Unit 2: - Entity Framework Core **(15)**

Introduction to Entity Framework Core, How to Install Entity Framework Core, DbContext in Entity Framework Core, Database Connection String in Entity Framework Core, CRUD Operations in Entity Framework Core, Entity States in Entity Framework Core, Data Annotation Attributes in Entity Framework Core- Table Attributes, Column Attributes, Key

Attribute, ForeignKey Attribute, Index Attribute, InverseProperty Attribute, NotMapped Attribute, Required Attribute, MaxLength and MinLength Attribute, Database Generated Attribute, TimeStamp Attribute, ConcurrencyCheck Attribute, Relationships in Entity Framework Core- One-to-One Relationships, One-to-Many Relationships, Many-to-Many Relationships, Self-Referencing Relationship, Asynchronous Programming with Entity Framework Core, Disconnected Entities in Entity Framework Core, Stored Procedures in Entity Framework Core, Transactions in Entity Framework Core, Migration in Entity Framework Core, Database Seedd in Entity Framework Core, Entity Framework Core Database First Approach

Unit 3: Model, View, Controller and Routing (20)

ViewData, ViewBag, Strongly Typed View, ViewModel, TempData, Post-Redirect-Get (PRG) Pattern Example, Layout View, Sections in Layout View, ViewStart, ViewImports, Partial Views, Different Ways to Render Partial View, View Components, Razor View Engine and Razor Syntax, How to Install and use Bootstrap in ASP.NET Core MVC, Action Results in ASP.NET Core MVC- Action Results, View Result, Partial View Result, JSON Result, Content Result, File Result, Redirect Results, Status Results, Object Result, EmptyResult , Routing in ASP.NET Core MVC, Custom Routing, Custom Route Constraints in Web Application, Attribute Routing, Attribute Routing using Tokens, Attribute Routing vs Conventional Routing, Model Binding in ASP.NET Core MVC, Model Binding using- FromForm, FromQuery, FromRoute, FromHeader, FromBody, Complex Type, Custom Model Binding in ASP.NET Core MVC

Unit 4: - HTML, Tag Helper, Data Annotation Validation and State management (15)

HTML Helpers for-TextBox, TextArea, DropDownList, RadioButton, Check Box, ListBox, Password, Hidden, Custom HTML Helper in ASP.NET Core MVC, Creating Form Using HTML Helpers, Different Ways to Generate Links in ASP.NET Core MVC, Tag Helpers for- Image Tag , Environment Tag, Navigation Menus, Form Tag, Partial Tag, Creating Custom Tag Helper, View Component Tag Helper, Cache Tag Helper, Data Annotations, Model Validations, Data Annotation Attributes- Custom Data Annotation, Remote Validation, Blacklist and Whitelist Checks using Data Annotation, Displaying and Formatting Attributes, Real-Time Examples of Data Annotations in ASP.NET Core MVC, Cookies, Encrypt Cookies, Persistent vs Non-Persistent Cookies, Sessions, In-Memory vs Distributed Sessions, Differences Between Cookies and Sessions, Upload File, Restrict Uploaded File Size, Restrict Uploaded File Type, Save Uploaded file to Database, Display Images, Delete Images, Upload Multiple Files, Export Data to Excel File, Import Excel Data to Database, Generate PDF, Generate Password Protected PDF, Convert HTML to PDF, Send Email with Attachment

Course Outcomes: -

Students will be able to:

1. Demonstrate the creation of ASP.NET Core MVC Web Applications using .NET 8.
2. Understand project file structure and implement Models, Controllers, Views, and Dependency Injection in ASP.NET Core MVC applications.
3. Utilize Entity Framework Core for Data Access and perform database operations using Entity Framework Core.

4. Discuss transactions, migration, and database seeding in Entity Framework Core.
5. Implement Reusability in Views using Partial Views, View Components, and the Razor View Engine for efficient view management.
6. Cover Action Results, Routing, Model Binding, HTML Helpers, and Tag Helpers in ASP.NET Core MVC applications.
7. Explain Data Annotations and Model Validations, including custom validations and remote validation.
8. Discuss different methods of State Management, including Cookies and Sessions.

Reference Books:

1. Pro ASP.NET Core MVC by ADAM FREEMAN, Apress
2. ASP.NET MVC with Entity Framework and CSS by Lee Naylor, Apress
3. ASP.NET MVC Core 2.0 Cookbook by Engin Polat & Stephanen Belkheraz, Packt Publishing;
4. ASP.NET Core in Action by Andrew Lock
5. ASP.NET Core Application Development: Building an application in four sprints (Developer Reference) by James Chambers, David Paquette & Simon Timms, Microsoft Press
6. Pro ASP.NET Core MVC by ADAM FREEMAN, Springer Nature
7. Professional ASPNET MVC 5 by Galloway Jon. Matson David. Wilson Brad. Allen K Scott, wiley

B.Sc. (Computer Science)-III (CBCS)							
Sem:	VI						
Paper Type:	DSE 2 **A (Elective)**			**Paper No:**	Paper XVI		
Paper Name:	Advanced Python						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. Understand the fundamentals of GUI programming and Gain proficiency in utilizing Tkinter, a GUI library in Python, for building graphical user interfaces.
2. Develop skills in managing GUI components, event handling, layout management, and customization of GUI elements.
3. Install and configure MySQL database software and MySQL Connector for Python.
4. Understand the concept of web frameworks and the Model-View-Controller (MVC) design pattern.
5. Develop web applications using Django, including creating views, defining URL routes, working with templates, and managing models and forms.
6. Introduction to Pandas library for data manipulation, data cleaning, preprocessing, aggregation, and summarization.
7. Explore data visualization techniques using Matplotlib and Seaborn libraries, including creating various types of plots and customizing them.
8. Introduction to NumPy for scientific computing, including array creation, manipulation, and mathematical operations.

Unit 1: - Windows Applications using Tkinter (10)

GUI Programming: GUI in Python, Advantages of GUI, Introduction to GUI library, Basic Operations using Tkinter, Root Window, Working with Containers: Frame, Canvas, Layout Management, Events and Bindings, Font, Colors, drawing on Canvas (line, oval, rectangle, etc.), Widgets: Label, Button, Checkbutton, Entry, Listbox, Message, Radiobutton, Text, Spinbox, Scrollbar, Menu etc., different dialog boxes and message boxes, Writing Python Programs for GUI applications

Unit 2: - Database Connectivity using MySQL (10)

Installation of MySQL Database Software, Installing MySQL Connector, Steps for Database Connectivity, Working with MySQL Database: Inserting, Retrieving, Deleting and Updating the data Working with Stored Procedure

Unit 3: - Web Application using Django **(20)**

What Is a Web Framework? The MVC Design Pattern, Django's History, Advantages of Django, Understanding Django environment, Installing Django, Setting Up a Database Django architecture, The Development Server, Django Commands Overview, Starting a Project, Django apps, Difference between app and project, The Project Structure, Setting Up Your Project, Create an Application, Migration, Admin Panel. Views in Django, URL Routing, Template in Django, Models in Django, Forms in Django.

Unit 4: - Data science using python **(20)**

Data Manipulation with Pandas- Introduction to Pandas library, Series and DataFrames, Creating Data Frame from an Excel Spreadsheet, from .csv file, from python Dictionary, from python List ,Tuples, Operations on Data Frames, Data cleaning and preprocessing, Data aggregation and summarization, Data Visualization with Matplotlib and Seaborn- Introduction to Matplotlib and Seaborn, installation, Creating basic plots: line plots, scatter plots, bar plots, Histogram ,a pie chart etc. Customizing plots and adding annotations, Exploratory data analysis through visualization, Introduction to NumPy, Overview of NumPy and its importance in scientific computing, Installing NumPy and setting up the environment, NumPy arrays: creation, indexing, slicing, and manipulation, Basic mathematical operations with NumPy arrays, Array Manipulation and Shape Manipulation- Reshaping arrays: reshape, resize, flatten, Concatenating and splitting arrays, Changing array dimensions. Transposing arrays

Course Outcomes: -

Students will be able to:

1. Demonstrate proficiency in utilizing Tkinter, a GUI library, to create graphical user interfaces in Python.
2. Develop skills in managing GUI components, including event handling, layout management, and customization of GUI elements.
3. Install and configure MySQL database software and MySQL Connector for Python.
4. Establish database connectivity and perform basic operations such as insertion, retrieval, deletion, and updating of data.
5. Understand the concept of web frameworks and the Model-View-Controller (MVC) design pattern.
6. Create web applications using Django, including defining views, URL routes, and working with templates.
7. Utilize the Pandas library for data manipulation tasks such as data cleaning, preprocessing, aggregation, and summarization.
8. Create various types of plots and customize them to effectively communicate insights from data using Matplotlib and Seaborn libraries.
9. Gain an understanding of NumPy for scientific computing, including array creation, manipulation, and mathematical operations.

Reference Books:

1. MySQL for Python: Database Access Made Easy- A. Lukaszewski
2. Beginning Django: Web Application Development and Deployment with Python-Daniel Rubio-Apress
3. Django Unleashed- Andrew Pinkham-SAMS
4. Practical Django Projects- James Bennett-Apress
5. Python GUI Programming with Tkinter- Alan D. Moore-Packt
6. Tkinter GUI Application Development H TSHOT - Bhaskar Chaudhary -Packt
7. Python for Data Analysis by Wes McKinney
8. Python Data Science Handbook by Jake VanderPlas

B.Sc. (Computer Science)-III (CBCS)							
Sem:	VI						
Paper Type:	DSE 2 **B (Elective)**			**Paper No:**	Paper XVI		
Paper Name:	Mobile Application Development using Kotlin						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. Student will understand Set up of Android Studio and understand a development environment for Android app development using Kotlin.
2. Students will able to Create interactive and user-friendly UIs by designing layouts, using XML, and incorporating UI elements such as buttons, text fields, and lists.
3. Students will able to understand user interaction by handling touch events, button clicks, and using dialogs to provide feedback and information.
4. Students will able to Develop multi-screen apps by understanding the concepts of activities, intents, and fragments for effective navigation
5. Student understands data passing between screens using intents, bundles, and fragment communication.
6. Students will learn about data storage options in Android, including SharedPreferences for simple data storage and SQLite databases for more complex data management.
7. Student will Understand the process of preparing and deploying an Android app to the Google Play Store.

Unit 1: - Introduction to Android App Development (20)

Overview of Android app development and Android Architecture, Setting up Android Studio., Creating a simple "Hello World" app., Understanding the Android project structure., User Interface (UI) Design and Layouts, UI Design Principles-Material Design guidelines, UI best practices, Layouts and Views and view groups-Introduction to Android layouts, Creating layouts in XML, Designing Interactive views-Buttons, text fields, labels, list, checkbox, spiner, radio and radio group, Handling user interaction, Building Complex UIs-Lists and RecyclerView, Custom UI components

Unit 2: - User Interaction and Navigation (15)

Handling User Input-Responding to button clicks and touch events, Toast messages and dialogs, android ap components activity, service, content provider, broadcost receiver, Navigating Between components, Creating multiple screens in your app, Fragment-Based UI, Introduction to Fragments, Building flexible UIs with fragments, Passing Data Between Screens, Sending data with Intents, Fragment communication

Unit 3: - Data Storage and Retrieval **(15)**

Introduction to Data Storage, Overview of data storage options in Android, Choosing the right storage method, Shared Preferences, Using Shared Preferences for simple data storage, Saving and retrieving data, SQLite Database, Introduction to SQLite databases in Android, Performing database operations, Integrating Data Storage, Incorporating data storage into your app, Best practices for data handling

Unit 4: - App Deployment and Finalization **(10)**

App Deployment, Preparing the app for the Google Play Store, Generating a signed APK, App Finalization, App optimization and performance, Gathering user feedback and making improvements, App Presentation, Final app project presentations, Sharing experiences and challenges, Creating SMS sending application, email application, google map application

Course Outcomes

Students will be able to:

1. Understand the setup process of Android Studio and establish a development environment for Android app development using Kotlin.
2. Create interactive and user-friendly user interfaces (UIs) by designing layouts using XM and incorporate various UI elements such as buttons, text fields, and lists to enhance the user experience.
3. Implement responsive UI elements that enhance user engagement and interaction.
4. Implement effective navigation between screens to provide seamless user experiences within the app.
5. Understand and implement data passing between screens using intents, bundles, and fragment communication.
6. Implement data storage solutions that suit the requirements of the app and ensure data integrity and security.
7. Learn the steps involved in deploying the app to the Google Play Store and ensure compliance with store guidelines and policies.

Reference Books: -

1. "Android App Development" by Reto Meier
2. "Kotlin for Android App Development" by Peter Sommerhoff
3. "Android User Interface Design: Turning Ideas and Sketches into Beautifully Designed Apps" by Ian G. Clifton
4. "Android Programming for Beginners" by John Horton
5. "Android Fragments" by Dave MacLean
6. "Android Programming: The Big Nerd Ranch Guide" by Bill Phillips and Chris Stewart
7. "Android SQLite Essentials" by Sunny Kumar Aditya
8. "Mastering Android Development with Kotlin" by Milos Vasic

B.Sc. (Computer Science)-III (CBCS)							
Sem:	VI						
Paper Type:	DSE 2 **C (Elective)**			**Paper No:**	Paper XVI		
Paper Name:	Data Visualization using Power BI						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective: -

Students will try to learn:

1. To gain proficiency in navigating the Power BI interface, understanding its components, and utilizing its basic features for data visualization.
2. To learn how to import, clean, and transform data from various sources into formats suitable for visualization in Power BI.
3. To understand the importance of data modeling techniques such as creating relationships, measures, and calculated columns for effective analysis.
4. To master the creation of various types of visualizations, including bar charts, line charts, scatter plots, and maps, using Power BI's intuitive tools.
5. To explore customization options to enhance visual appeal and effectively convey insights to stakeholders.
6. To learn to design interactive dashboards by combining multiple visualizations, slicers, filters, and drill-down functionalities.
7. To understand principles of dashboard layout, organization, and storytelling to create engaging and actionable presentations of data.
8. To understand the process of sharing Power BI reports and dashboards securely with colleagues and stakeholders.

Unit 1: - Introduction to Power BI **(10)**

Overview of Business Intelligence, BI Uses and Users, Various BI Tools, Why Power BI, Introduction to Power BI, Features of Power BI, Power BI Components, Building Blocks of Power BI, Architecture of Power BI, Power BI Desktop Installation, Loading and Transforming dataset: Data Sources-File Sources, Databases, Azure, Other Sources, Loading Data-Web Pages, CSV Files, Text Files, XML Files, Excel, Microsoft Access Databases, SQL Server and other databases, Refreshing Data.

Unit 2: - Creating a Data Model **(10)**

Data Modeling in the Power BI Desktop Environment-The Power BI Desktop Data View, Data Models, Managing Power BI Desktop Data- Manipulating Tables, Manipulating Columns, Power BI Desktop Data Types, Formatting Power BI Desktop, Data Currency Formats, Preparing Data for Dashboards, Categorize Data, Apply a Summarization, Define Sort by Columns, Sorting Data in Power BI Desktop Tables, Adding Hierarchies, Designing a Power BI Desktop Data Model- Data View and Relationship View, Creating and Deleting Relationships Manually and Automatically

Unit 3: - Transforming Datasets **(20)**

Editing Data After a Data Load, Transforming Data Before Loading, Dataset Shaping- Renaming Columns, Reordering Columns, Removing Columns, Merging Columns, Duplicating Columns, Splitting Columns, Removing Records, Removing Duplicate Records, Sorting Data, Reversing the Row Order, Filtering Data-Selecting Specific Values, Finding Elements in the Filter List, Filtering Text Ranges, Filtering Numeric Ranges, Filtering Date and Time Ranges, Data Cleansing: Viewing a Full Record, Changing Data Type , Detecting Data Types, Replacing Values, Transforming Column Contents, Filling Down, Using the First Row As Headers, Grouping Records, Extending Data, Appending Data, Merging Data- Adding Data, Aggregating Data During a Merge Operation, Extending the Data Model with Calculated Columns, Creating Custom Columns, Index Columns, Types of Join- Joining on Multiple Columns, Preparing Datasets for Joins, Correct and Incorrect Joins, Examining Joined Data, The Expand and Aggregate Buttons.

Unit 4: - Power Query Editor and Data Visualizations **(20)**

Power Query Editor - What is DAX, Different type of DAX functions-Aggregate functions, Date functions, Logical functions, Math functions, String functions, Trigonometric functions and other functions. Adding Measures to the Data Model Basic Aggregations in Measures, Using Multiple Measures, Cross-Table Measures, More Advanced Aggregations, Filtering Data in Measures, Analyzing Data over Time. Data Visualizations Charts in Power BI-Types of charts, Maps in Power BI, Table and Matrix in Power BI, Subtotal and Total in Matrix, Cards and Filters in Power BI, Conditional Formatting, Slicers in Power BI- slicers, adding a Slicer, Applying Slicers, clearing a Slicer, deleting20 34 / 52 a Slicer, modifying a Slicer, Formatting Slicers-Slicer Orientation, Modifying the Outline, Adjusting Selection Controls, Setting the Exact Size and X and Y coordinates of a Slicer, Slicer Header, Slicer Items Designing Power BI Dashboards and Reports Dashboards, reports, Dashboards versus reports, Dashboard design- What is KPI, Whento use KPI, Requirements for KPI, KPI Visualizations, Visual selection, Layout, Navigation pane, Full screen mode, Supporting tiles, Custom date filters, Single- dashboard, Multiple-dashboard, Organizational dashboards, Multiple datasets Dashboard tiles- Tile details and custom links, Images and text boxes, SQL Server Reporting Services. Deploying the Power BI Report Server Live Dashboard pages, Live report pages, Mobile-optimized dashboards Case study – Superstore, IPL Analysis, Product Sales Data Analysis, Marketing Campaign Insights Analysis, Financial Performance Analysis, Loan Application Analysis

Course Outcomes: -

Students will be able to:

1. Developed proficiency in navigating the Power BI interface, comprehending its components, and employing fundamental features for effective data visualization.
2. Gain skills to import, cleanse, and transform data from diverse sources, ensuring its suitability for visualization within Power BI.
3. Gain insight into the significance of data modeling strategies, encompassing the establishment of relationships, formulation of measures, and creation of calculated columns to facilitate insightful analysis.
4. Demonstrate mastery in generating various visualization types, such as bar charts, line charts, scatter plots, and maps, utilizing the user-friendly tools available in Power BI.
5. Explore a spectrum of customization options to elevate the visual allure of presentations and effectively communicate insights to stakeholders.
6. Develop proficiency in crafting interactive dashboards by amalgamating diverse visualizations, slicers, filters, and drill-down functionalities to enhance user engagement and comprehension.
7. Comprehend principles governing dashboard layout, organization, and narrative construction, enabling the creation of captivating and actionable data presentations.
8. Understand the protocol for securely sharing Power BI reports and dashboards with colleagues and stakeholders, fostering efficient collaboration and knowledge dissemination.

Reference Books: -

1. Pro Power BI Desktop-Free interactive data analysis with Microsoft Power BI by AdamAspin, Apress
2. Introducing Microsoft Power BI by Alberto Ferrari and Marco Russo, Microsoft Press
3. Mastering Microsoft Power BI by Brett Powell, Packt BIRMINGHAM– MUMBAI
4. Microsoft Power BI Complete Reference by Devin Knight, Brian Knight, Mitchell Pearson, Manuel Quintana, Brett Powell, Packt
5. Learn Power BI by Greg Deckler,Packt
6. Pro Power BI Desktop-Free interactive data analysis with Microsoft Power BI by AdamAspin,Apress

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V						
Paper Type:	SEC			**Paper No:**	SEC-2		
Paper Name:	Software Testing						
Credit:	04			**Theory:**	4 Hrs./week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Course Objective:-

Students will try to learn:

1. Basic software debugging methods.
2. White box testing methods and techniques.
3. Black Box testing methods and techniques.
4. Designing test plans.
5. Different testing tools (familiar with open source tools)

Unit 1: - Introduction To Software Testing (10)

What is Software Testing? Use or need of software testing. ,Software Development Life Cycle (SDLC) :- Water Fall Model, Spiral Model, V- Model, Prototype Model, Hybrid Model

Unit 2: - White Box and Black Box Testing (20)

Introduction to White box testing, Advantages and Disadvantages of White box testing, Loop Testing, Path Testing , Condition testing , Memory Testing , Performance Testing, Black Box Testing:- Introduction to black box testing , Advantages and Disadvantages of black box testing, **Functional Testing-** Integration Testing (Incremental Integration Testing) ,Top Down Incremental Integration Testing , Bottom Up Incremental Integration Testing , Non Incremental Integration Testing , System Testing , Acceptance Testing , Smoke Testing , Exploratory Testing , Adhoc Testing , Performance Testing – Load Testing, Stress Testing, Volume Testing, Soak Testing, Regression Testing-Unit Regression Testing/Retest, Regional Regression Testing, Full Regression Testing

Unit 3: - Test cases and its design Techniques: (15)

Introduction to Test Case , Characteristics Of Good Test Case , Test Case Template, How To Write A Test Case, How To Ensure The Test Coverage Is Good , How To Identify whether It Is a Good Test Case Or Not, Review Process/Peer Review , Preparing Review Report, Examples On Writing Test Cases, Test Cases Design Techniques-Error Guessing, Equivalence Partitioning, Boundary

Value Analysis

Unit 4: - Software Test Life cycle and Defect Life Cycle: (15)

Software Test Life Cycle-Writing Test Plan, Preparing Traceability Matrix, Writing Test Execution Report, Summary Report, Retrospect Meeting /Triage Meetings, Defect Life Cycle-Concept of Defect life cycle, Difference between Bug, Defect, Failure, Error

Course Outcomes:-

Students will able to:

1. Investigate the reason for bugs and analyze the principles in software testing to prevent and remove bugs.
2. Implement various test processes for quality improvement
3. Design test planning.
4. Manage the test process
5. Use practical knowledge of a variety of ways to test software and an understanding of some of the tradeoffs between testing techniques.

Reference Books:

1. The art of Software Testing– Glenford J. Myers
2. Lessons learned in Software Testing – CemKaner, James Bach, Bret Pettichord
3. A Practitioner's Guide to Software Test Design- Lee Copeland
4. Software Testing Techniques, 2nd edition- Boris Beizer
5. How to Break Software: A Practical Guide to Testing- James Whittaker

B.Sc. (Computer Science)-III (CBCS)						
Sem:	V & VI					
Paper Type:	Practical		**Paper No:**	Practical V		
Paper Name:	Practical based on DSC 1F & 1I					
Credit:	04		**practical:**	5 Hrs./ Week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100

Practical based on DSC 1F (Advanced Java)

1. Write a java socket programming in which client sends a text and server receives it.
2. Write a program to demonstrate URL class.
3. Write a program to demonstrate InetAddress class.
4. Write a program to demonstrate use of Datagram Socket.
5. Write a program to create Student registration form using Swing Component.
6. Write the following program using Swing component. An Election is conducted between 3 candidates. There are N number of voters. By clicking Next Voter Button textboxes and RadioButtons need to be cleared. By clicking Results, the votes obtained by each candidate and the winner candidate to be displayed in text area. Exit button should exit program.
7. Write a program for inserting, updating and deleting data into/from table using PreparedStatement.
8. Write a program to demonstrate callable statement.
9. Write a Servlet program to check that life cycle methods are called by web container.
10. Write a program to create simple servlet for displaying welcome message.
11. Write a program to create servlet for session management using cookies.
12. Write a program to create servlet for session management using Hidden Form Field.
13. Write a program to create servlet for session management using URL Rewriting.
14. Write a simple program of authenticating user using filter.
15. Write a simple program to demonstrate the use of request dispatcher.
16. Write a simple program to demonstrate the use of Send Redirect.
17. Write a JSP program to count number of visitors.
18. Write a program for communication between HTML & JSP.
19. Set up a Hibernate configuration file for a project and configure it to connect to a database. Create a Hibernate mapping for a simple entity class and map it to a database table.
20. Implement CRUD operations (Create, Read, Update, Delete) using Hibernate's Session API.
21. Implement Hibernate annotations for mapping entities instead of XML mapping files.

22. Integrate Hibernate with Spring framework and configure session management using Spring's transaction management.
23. Set up a basic Spring project and configure dependencies in the pom.xml or build.gradle file.
24. Implement Dependency Injection (DI) and Inversion of Control (IoC) using Spring's @Autowired annotation.
25. Use Spring Data JPA to perform database operations with CRUD repositories.

Practical based on DSC 1 I (Linux and Shell Programming)

1. Navigate the file system using commands such as ls, cd, pwd, mkdir, and rmdir.
2. View file contents using commands such as cat, less, and head.
3. Manipulate files and directories using commands such as cp, mv, rm, and touch.
4. Use commands such as chmod and chown to change file permissions and ownership.
5. Understand the concept of file permissions (read, write, execute) for users, groups, and others.
6. View running processes and their resource usage using commands such as ps, top, and htop.
7. Manage processes using commands such as kill, killall, pkill, and pgrep.
8. Monitor system performance using commands such as df, du, and free.
9. Configure network settings using commands such as ifconfig, ip, and netstat.
10. Manage users and groups using commands such as useradd, usermod, groupadd, and passwd.
11. Shell Programming:
12. Write and execute a simple shell script using a text editor and the bash interpreter.
13. Use variables to store and manipulate data within a shell script.
14. Write comments to document the purpose and functionality of shell scripts.
15. Implement conditional statements (if-else) in shell scripts to control program flow based on different conditions.
16. Use loop constructs (for, while) to iterate over lists of items or perform repetitive tasks.
17. Define and call functions within shell scripts to encapsulate reusable code blocks.
18. Pass arguments to functions and return values using the return statement.
19. Organize shell scripts into modular components for better code organization and maintainability.
20. Read input from files and process data using commands such as cat, grep, awk, and sed.
21. Use redirection operators (>, >>, <) to redirect input and output streams between files and commands.
22. Implement error handling mechanisms in shell scripts to gracefully handle unexpected conditions.
23. Use exit codes and error messages to communicate errors to users and other scripts.
24. Write Linux script for checking given number is prime, Armstrong and palindrome.
25. Write Linux script to display Fibonacci sequence up to n numbers.

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V & VI						
Paper Type:	Practical			**Paper No:**	Practical VI		
Paper Name:	Practical based on DSC 1G & 1J						
Credit:	04			**practical:**	5 Hrs./ Week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Practical based on DSC 1G (Dot NET Core)

1. Create a simple console application that prints "Hello, World!" to the console.
2. Compile and run the application using the .NET Core CLI or Visual Studio.
3. Perform basic arithmetic operations (addition, subtraction, multiplication, division) on numeric variables.
4. Use string interpolation or concatenation to display variable values.
5. Implement conditional statements (if-else, switch-case) to control program flow based on different conditions.
6. Use loops (for, while, do-while) to iterate over arrays, collections, or sequences of data.
7. Create nested loops and conditional statements for more complex control flow logic.
8. Declare and initialize arrays of different data types.
9. Access array elements using index notation and perform array manipulation operations (sorting, searching, etc.).
10. Define and call methods with different access modifiers (public, private, protected).
11. Pass parameters to methods and return values from methods.
12. Overload methods with different parameter types and number of parameters.
13. Create classes and objects to represent real-world entities.
14. Create class library for checking Odd number, even number, prime number, Armstrong number etc.
15. Implement encapsulation, inheritance, and polymorphism concepts in C#.
16. Use constructors, properties, and methods to define the behavior of objects.
17. Implement try-catch blocks to handle exceptions and prevent application crashes.
18. Throw custom exceptions to handle specific error conditions.
19. Create custom exception and use them.
20. Use finally blocks to execute cleanup code regardless of whether an exception is thrown.
21. Read from and write to text files using StreamReader and StreamWriter classes.
22. Implement file input/output operations such as reading, writing, appending, and deleting files.

23. Handle file exceptions and ensure proper resource management using IDisposable interface.
24. Use LINQ queries to perform filtering, sorting, grouping, and aggregation operations on collections.
25. Create and manage multiple threads using the Thread class or ThreadPool.

Practical based on DSC 1J (ASP.NET Core MVC)

1. Create a new ASP.NET Core MVC project using Visual Studio or the .NET CLI.
2. Explore the project structure and understand the role of important files such as Startup.cs, Program.cs, and the Views folder.
3. Define model classes representing entities in the application domain.
4. Generate scaffolded controllers and views using Entity Framework Core for CRUD operations on the model classes.
5. Customize the generated views and controllers to meet specific requirements.
6. Define custom routes using attribute routing and convention-based routing.
7. Implement route constraints to restrict the format of URL parameters.
8. Deminstrate how routing works and how URLs map to controller actions.
9. Create HTML forms for user input and data submission.
10. Implement form validation using data annotations and ModelState.IsValid.
11. Bind form data to model properties using model binding techniques.
12. Set up a database context and configure entity classes for use with Entity Framework Core.
13. Perform database migrations to create or update the database schema based on changes to the model
14. Demonstrate HTML helper tag.
15. Demonstrate Tag helper.
16. Data Annotation Attributes and Relationships in Entity Framework Core
17. Perform CRUD operations using Stored Procedures in Entity Framework Core
18. Demonstrate migration concept and add sample record using seeding technique
19. Demonstrate routing concept in ASP.Net core MVC
20. Demonstrate different ActionResult concept.
21. Implement CRUD operations (Create, Read, Update, Delete) using Entity Framework Core methods.
22. Create RESTful API endpoints for accessing application data using ASP.NET Core MVC controllers.
23. Implement HTTP methods (GET, POST, PUT, DELETE) to perform CRUD operations on resources.
24. Use attribute routing and model binding to define API routes and handle incoming requests.
25. Validate form input and handle form submissions using JavaScript before sending requests to the server.

B.Sc. (Computer Science)-III (CBCS)							
Sem:	V & VI						
Paper Type:	Practical			**Paper No:**	Practical VII		
Paper Name:	Practical based on DSE 1 A/B/C & 2 A/B/C						
Credit:	04			**Practical:**	5 Hrs./ Week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100	

Practical based on DSE 1 A (Python Programming)

1. Write a simple Python script that prints "Hello, World!" to the console.
2. Perform basic arithmetic operations (addition, subtraction, multiplication, division) on numeric variables.
3. Concatenate strings and convert between data types as needed.
4. Implement conditional statements (if-else, elif) to control program flow based on different conditions.
5. Use loops (for, while) to iterate over lists, tuples, dictionaries, and ranges.
6. Create nested loops and conditional statements for more complex control flow logic.
7. Create and manipulate lists, tuples, dictionaries, and sets.
8. Access elements in data structures using indexing and slicing operations.
9. Use list comprehensions and generator expressions for concise data manipulation.
10. Define and call functions with different numbers of parameters.
11. Use default arguments, variable-length argument lists, and keyword arguments in function definitions.
12. Return values from functions and handle function exceptions using try-except blocks.
13. Read from and write to text files using built-in file handling functions (open, read, write, close).
14. Parse CSV files using the csv module and manipulate data as needed.
15. Handle file exceptions and ensure proper resource management using context managers (with statement).
16. Define classes with attributes and methods to represent real-world entities.
17. Implement encapsulation, inheritance, and polymorphism concepts in Python.
18. Use constructors, properties, and method overriding to define the behavior of objects.
19. Implement try-except blocks to handle exceptions and prevent program crashes.
20. Throw custom exceptions to handle specific error conditions and provide meaningful error messages.
21. Use finally blocks to execute cleanup code regardless of whether an exception is raised.
22. Create custom modules to encapsulate reusable code and import them into other Python scripts.
23. Organize related modules into packages and import them using absolute and relative import

statements.

24. Create abstract class and interface and implement them.
25. Overload any 10 operators.

Practical based on DSE 1 B (Kotlin Programming)

1. Compile and run the program using the Kotlin compiler or an Integrated Development Environment (IDE) like IntelliJ IDEA.
2. Declare variables of different data types (Int, Float, Double, String, Boolean).
3. Perform basic arithmetic operations (addition, subtraction, multiplication, division) on numeric variables.
4. Implement conditional statements (if-else, when) to control program flow based on different conditions.
5. Use loop structures (for, while, do-while) to iterate over arrays, lists, or ranges of values.
6. Create nested loops and conditional statements for more complex control flow logic.
7. Define and call functions with different parameters and return types.
8. Implement higher-order functions that take other functions as parameters or return functions as results.
9. Use lambda expressions to define anonymous functions and pass them as arguments to other functions.
10. Create and manipulate collections such as lists, sets, maps, and arrays in Kotlin.
11. Perform common operations on collections, such as iteration, filtering, mapping, and folding.
12. Use Kotlin's standard library functions (e.g., filter, map, reduce) to process collections efficiently.
13. Define classes and objects to represent real-world entities in Kotlin.
14. Implement inheritance, polymorphism, and encapsulation principles in Kotlin classes.
15. Use data classes and object declarations to create simple data structures and singleton objects.
16. Understand Kotlin's null safety features and how to handle nullable types using safe calls (?.), null checks, and the elvis operator (?:).
17. Use the safe cast operator (as?) and smart casts to safely cast nullable types to non-nullable types.
18. Create extension functions and extension properties to add new functionality to existing classes without modifying their source code.
19. Overload operators such as plus (+), minus (-), times (*), and equals (==) to define custom behavior for user-defined types.
20. Read from and write to files using Kotlin's standard library functions for file I/O.

21. Handle exceptions and errors using try-catch blocks and the throw keyword to gracefully handle unexpected conditions in Kotlin programs.
22. WAP which will demonstrate use of abstract class.
23. WAP to demonstrate interface and Multiple interfaces in kotlin.
24. WAP to implement following data classes-
    a. equals (), b. Hash Code () c. To String () d. Copy ().
25. WAP to which demonstrate all exception handler

Practical based on DSE 1 C (Data warehousing and Mining)

1. What options are available on main panel? Describe and create the arff and csv file format.
2. Load iris dataset. How many instances are this dataset have? How many attributes? What is the range of possible values of the attribute petallength?
3. Explore the dataset using various visualization tools available in Weka.
4. Analyze basic statistics of the dataset such as mean, median, standard deviation, etc.
5. Identify any missing values and outliers in the dataset.
6. Preprocess the dataset by handling missing values, outliers, and noise.
7. Perform attribute selection to identify the most relevant features.
8. Normalize, discretize or standardize numerical attributes.
9. Load the weather.nominal dataset. Use the following filters in weka.
    a. unsupervised.instance.RemoveWithValues to remove all instances in which the humidity attribute has the value high.
    b. Convert numeric value to nominal
    c. Convert nominal to string
    d. Discretizes data
10. Load the iris dataset. Use the following filter of weka.
    a. Add noise to last column (i.e. Class).
    b. Randomize the data
    c. Normalize the data
    d. Reorder the data
11. Implement and evaluate different classification algorithms available in Weka (e.g., Decision Trees, Naive Bayes, k-Nearest Neighbors).
12. Train the models using the training set and evaluate their performance on the testing set using various metrics like accuracy, precision, recall, and F1-score.

13. Compare the performance of different classifiers and identify the most suitable one for the dataset.
14. Apply regression algorithms available in Weka (e.g., Linear Regression, Polynomial Regression).
15. Train the regression models using the training set and evaluate their performance on the testing set using metrics like Mean Squared Error (MSE) and R-squared.
16. Implement and evaluate clustering algorithms available in Weka (e.g., k-Means, Hierarchical Clustering).
17. Explore different clustering techniques and their impact on clustering quality.
18. Visualize the clusters using Weka's visualization tools and analyze their characteristics.
19. Apply association rule mining algorithms (e.g., Apriori) to identify interesting patterns in the dataset.
20. Adjust parameters such as minimum support and confidence to control the quality of the discovered rules.
21. Interpret and analyze the discovered rules to gain insights into the dataset.
22. Implement feature selection techniques to identify the most informative features in the dataset.
23. Evaluate the impact of feature selection and dimensionality reduction on the performance of classification or clustering algorithms.
24. Implement ensemble learning techniques (e.g., Random Forest, AdaBoost) using Weka's ensemble classifiers.
25. Compare the performance of ensemble methods with individual classifiers and analyze the benefits of ensemble learning.

Practical based on DSE 2 A (Advanced Python)

1. Build a simple application with Tkinter that includes widgets such as labels, buttons, entry fields, and text areas and arrange widgets using different geometry managers (pack, grid, place) to create the desired layout. Validate user input and provide feedback for invalid entries using message boxes or labels.
2. Design a simple calculator application using Tkinter with buttons for numeric input and arithmetic operations.
3. Develop a CRUD (Create, Read, Update, Delete) application with Tkinter for managing a simple database-backed model.
4. Develop application which display employee records in tabular format.
5. Extend Tkinter by creating custom widgets with unique functionality or appearance. Implement custom widgets such as color pickers, sliders, or progress bars using Tkinter's canvas or frame

widgets.

6. Design a menu-driven application with Tkinter that includes dropdown menus, cascading menus, and context menus.
7. Implement file dialogs, message boxes, and other common dialogs to interact with the user and perform actions such as opening, saving, and closing files.
8. Define URL patterns, views, and templates to render dynamic HTML pages using Django's templating engine.
9. Develop a CRUD (Create, Read, Update, Delete) application with Django for managing a simple database-backed model.
10. Create views and templates for listing, adding, editing, and deleting records from the database.
11. Implement form validation and error handling to ensure data integrity and user-friendly interactions.
12. Integrate user authentication and authorization into a Django application using Django's built-in authentication system. Create user registration and login forms, and implement password reset functionality. Restrict access to certain views or functionalities based on user roles and permissions.
13. Define models with relationships (one-to-one, one-to-many, many-to-many) to represent complex data structures. Implement CRUD operations for related models and navigate relationships using Django's ORM (Object-Relational Mapper).
14. Use Django's admin interface to manage database records and relationships.
15. Load a dataset into a Pandas DataFrame from a CSV file or other data sources.
16. Explore the dataset using descriptive statistics, such as mean, median, standard deviation, and quantiles. Perform data manipulation tasks, such as filtering rows, selecting columns, and sorting values. Handle missing values in the dataset by imputing, removing, or interpolating them also detect and remove duplicate rows or columns from the dataset.
17. Group data in the DataFrame based on one or more columns and compute summary statistics for each group. Aggregate data using functions such as sum, mean, median, min, max, and count.
18. Create NumPy arrays from Python lists or using built-in functions like np.array, np.zeros, np.ones, and np.random. Perform basic array operations, such as indexing, slicing, reshaping, and concatenation and apply element-wise mathematical operations to arrays, such as addition, subtraction, multiplication, and division.
19. Compute summary statistics for arrays, such as mean, median, sum, min, and max.
20. Aggregate data along specific axes or dimensions using functions like np.sum, np.mean, np.std, and np.percentile.
21. Create line plots, scatter plots, and bar plots using Matplotlib's pyplot interface and Customize plot

appearance by modifying plot attributes, such as colors, markers, labels, and titles. Add annotations, legends, and grid lines to enhance plot readability and interpretability.

22. Create multiple subplots within a single figure using Matplotlib's subplot and subplots functions. Customize subplot layouts, including the number of rows and columns and their arrangement.
23. Create visualizations for statistical analysis, such as distribution plots (histograms, kernel density estimates), box plots, and violin plots.
24. Visualize relationships between variables using scatter plots, pair plots, and joint plots. Generate categorical plots, such as bar plots, count plots, and point plots, to compare different categories of data.
25. Customize Seaborn plots using built-in themes and color palettes to improve aesthetics and readability. Modify plot styles, fonts, and sizes to match specific visualization requirements. Customize plot elements, such as axis labels, tick marks, and legends, to enhance interpretability.

Practical based on DSE 2 B (Mobile app development using Kotlin)

1. Set up a new Android project in Android Studio with Kotlin support.
2. Create a simple "Hello World" app that displays a greeting message on the screen. Run the app on an emulator or physical device to verify functionality.
3. Design and implement a user interface with various UI components such as TextViews, EditTexts, Buttons, and ImageViews.
4. Create an application that takes the name from a text box and shows hello message along with the name entered in text box, when the user clicks the OK button
5. Use layout files (XML) to define the structure and appearance of the app's screens. Apply styling and theming to enhance the visual appeal of the app.
6. Create a screen that has input boxes for User Name, Password, Address, Gender(radio buttons for male and female), Age (numeric), Date of Birth (Date Picket), State (Spinner) and a Submit button. On clicking the submit button, print all the data below the Submit Button (use any layout)
7. Implement event listeners to handle user interactions with UI elements (e.g., button clicks, text input).
8. Validate user input and provide feedback to users for incorrect or incomplete input.
9. Use input controls such as EditTexts and Spinners to capture user data.
10. Implement navigation between different screens or fragments within the app.
11. Use activities, fragments, and intents to navigate between screens and pass data between them.
12. Create a multi-screen app with a navigation drawer, tab layout, or bottom navigation bar for seamless navigation.

13. Integrate data storage solutions such as SharedPreferences, SQLite databases, or Room Persistence Library to store and retrieve app data.
14. Implement CRUD (Create, Read, Update, Delete) operations for managing data records within the app.
15. Fetch and display remote data from RESTful APIs using libraries like Retrofit or Volley.
16. Handling Device Features:
17. Design an android application Send SMS using Intent
18. Design an android application for menu.
19. Access and utilize device features such as camera, location, sensors, and permissions.
20. Implement functionality to capture photos or videos using the device's camera.
21. Retrieve the user's current location and display it on a map using Google Maps API.
22. Prepare the app for deployment by optimizing performance, reducing APK size, and ensuring compliance with platform guidelines.
23. Create a user registration application that stores the user details in a database table.
24. Generate signed APKs or app bundles for release and upload them to Google Play Store or other app distribution platforms.
25. Manage app releases, updates, and user feedback through the app store console and monitoring tools.

Practical based on DSE 2 C (Data Visualization using Power BI)

1. Import data from different sources such as Excel, CSV, databases, and web services into Power BI.
2. Perform data cleaning tasks such as removing duplicates, handling missing values, and formatting data types.
3. Apply data transformation operations such as splitting columns, merging queries, and creating calculated columns.
4. Create relationships between different tables in the dataset.
5. Define hierarchies and drill-down paths to facilitate data exploration.
6. Implement calculated measures and calculated columns using DAX (Data Analysis Expressions).
7. Design interactive reports and dashboards using a variety of visualization types (e.g., bar charts, line charts, pie charts, maps, tables).
8. Customize the appearance of visualizations by adjusting formatting options such as colors, fonts, and labels.
9. Demonstrate at list 5 Maps
10. Demonstrate Table and Matrix with total and subtotal.

11. Demonstrate drill operations on any visuals
12. Incorporate slicers, filters, and bookmarks to enable users to interactively explore data.
13. Implement advanced calculations using DAX functions (e.g., CALCULATE, FILTER, RELATED, SUMMARIZE).
14. Perform time intelligence calculations such as year-over-year growth, moving averages, and cumulative totals.
15. Utilize AI-powered features such as Q&A (natural language querying) and Quick Insights to uncover hidden patterns and trends in data.
16. Create interactive tooltips and drill-through pages to provide additional context and detail to visualizations.
17. Demonstrate at list 5 Maps
18. Demonstrate any 10 visuals with formatting.
19. Demonstrate how to insert different objects and adding action to them.
20. Demonstrate different levels of filters.
21. Design sample report and dashboard.
22. Implement dynamic filtering and highlighting to focus attention on relevant data points.
23. Publish reports and dashboards to the Power BI Service for sharing with colleagues and stakeholders.
24. Configure row-level security to restrict access to sensitive data based on user roles and permissions.
25. Explore and utilize custom visuals available in the Power BI marketplace to enhance data visualizations.

B.Sc. (Computer Science)-III (CBCS)						
Sem:	V & VI					
Paper Type:	Practical		**Paper No:**	Practical IV		
Paper Name:	Project					
Credit:	04		**practical:**	5 Hrs./ Week		
Marks:	**UA:**	80	**CA:**	20	**Total:**	100

General Instructions Regarding Preparation Of Project Report For B.Sc. (Computer Science)-III Sem-VI

Typing

- The typing shall be standard 12 pts in double spaced
- Margins must be Left 2 inches Right 1.5 inches, Top 2 inches Bottom 1.5 inches
- Paper A4 size Bond Paper
- Report must be at least 50 pages

Copies

- Two hard-bound copies (Black Rexene with Golden Embossing as per format displayed herewith)- One original and one clean Xerox Copy.

Format For Title Page And For Embossing

PROJECT REPORT

ON

TITLE OF THE SYSTEM

Submitted in partial fulfillment for the award of degree of

BACHLER IN SCIENCE

IN

COMPUTER SCIENCE

To the

Punyashlok Ahilyadevi Holkar Solapur University, Solapur

Submitted By

NAME OF STUDENTS

Name of College

Department of Computer Science

Under The Guidance

Name of Guide

20___ - 20__

The Guidelines regarding the documentation and scope of project are mentioned here below:

For Application Level Projects

Project Report should be submitted in following format for Commercial Application Projects viz. Payroll, Sales, Purchase, Inventory, Book Shop, Examination system etc.

- Blank Pages at beginning
- Title Page
- Certificate Page
- Certificate from Guide and Head of Department
- Declaration
- Acknowledgement Page
- Index with printed Page with Numbers

1. CHAPTER 1: INTRODUCTION
    1.1. Existing System and Need for System
    1.2. Scope of Work
    1.3. Operating Environment – Hardware and Software
    1.4. Detail Description of Technology Used
2. CHAPTER 2: PROPOSED SYSTEM
    2.1. Proposed System
    2.2. Objectives of System
    2.3. User Requirements
3. CHAPTER 3 : ANALYSIS & DESIGN
    3.1. Data Flow Diagram (DFD)
    3.2. Entity Relationship Diagram (ERD)
    3.3. Data Dictionary
    3.4. Table Design
    3.5. Code Design
    3.6. Menu Tree
    3.7. Input Screens
    3.8. Report Formats
    3.9. Test Procedures and Implementation
4. CHAPTER 4: USER MANUAL
    4.1. User Manual

    4.2. Operations Manual / Menu Explanation
    4.3. Forms and Report Specifications
    4.4. Drawbacks and Limitations
    4.5. Proposed Enhancements
5. CHAPTER 5: Conclusions
6. Bibliography
7. ANNEXURES :
    7.1. ANNEXURE 1 : INPUT FORMS WITH DATA
    7.2. ANNEXURE 2 : OUTPUT REPORTS WITH DATA
    7.3. ANNEXURE 3 : SAMPLE CODE

- Blank Pages at the end.

For System-Level Projects

Project Report should be submitted in following format for Commercial Application Projects viz. Payroll, Sales, Purchase, Inventory, Book Shop, Examination system etc.

- 2 Blank Pages at beginning
- Title Page
- Certificate from Company
- Certificate from Guide and Head of the Department
- Declaration
- Acknowledgement
- Index with printed Page Numbers

1. CHAPTER 1: INTRODUCTION
    1.1. Existing System and Need for System
    1.2. Scope of Work
    1.3. Operating Environment – Hardware and Software
    1.4. Detail Description of Technology Used
2. CHAPTER 2: PROPOSED SYSTEM
    2.1. Proposed System
    2.2. Objectives of System
    2.3. User Requirements

3. CHAPTER 3: ANALYSIS & DESIGN
    3.1. Object Diagram
    3.2. Class Diagram
    3.3. Use Case Diagrams
    3.4. Module Hierarchy Diagram
    3.5. Component Diagram
    3.6. Module Specifications
    3.7. Interface Diagram (in case of WAP and Embedded Systems)
    3.8. User Interface Design (Screens etc.)
    3.9. Table specifications (in case back end is a database)
    3.10. Test Procedures and Implementation
4. CHAPTER 4: USER MANUAL
    4.1. User Manual
    4.2. Operations Manual / Menu Explanation
    4.3. Program Specifications / Flow Charts
    4.4. Drawbacks and Limitations
    4.5. Proposed Enhancements
5. CHAPTER 5: Conclusions
6. Bibliography
7. ANNEXURES :
    7.1. ANNEXURE 1 : USER INTERFACE SCREENS
    7.2. ANNEXURE 2 : OUTPUT REPORTS WITH DATA (if any)
    7.3. ANNEXURE 3 : SAMPLE PROGRAM CODE (which will prove sufficient

- 2 Blank Pages at the end.