Savitribai Phule Pune University
(Formerly University of Pune)

Two Year Masters Degree Program in Computer Science
(Faculty of Science and Technology)

Syllabi for
M.Sc. (Computer Science) Part-II

(For Colleges Affiliated to Savitribai Phule Pune University)

Choice Based Credit System (CBCS) Syllabus
Under National Education Policy (NEP)

To be implemented from Academic Year 2024-2025

Savitribai Phule Pune University

Syllabus Structure as per NEP Guidelines

M. Sc. (Computer Science) Part-I from 2023-24

SEMESTER I

Course Type	Course code	Course Name	Credits		Teaching Scheme Hrs/Week		Examination Scheme and Marks		
			TH	PR	TH	PR	CE	EE	Total
Major Core	CS-501-MJ	Advanced Operating System	4	-	4	--	30	70	100
	CS-502-MJ	Artificial Intelligence	4	-	4	--	30	70	100
	CS-503-MJ	Principles of Programming Languages	2	-	2	--	15	35	50
	CS-504-MJP	Lab course on CS-501-MJ	-	2	--	4	15	35	50
	CS-505-MJP	Lab course on CS-502-MJ	-	2	--	4	15	35	50
Major Elective	CS-510-MJ	Advance Databases and Web Technologies	2	-	2	--	15	35	50
	CS-511-MJP	Lab course on CS-510-MJ	-	2	--	4	15	35	50
	OR								
	CS-512-MJ	Cloud Computing	2	-	2	--	15	35	50
	CS-513-MJP	Lab course on CS-512-MJ	-	2	--	4	15	35	50
	OR								
	CS-514-MJ	C# .NET Programming	2	-	2	--	15	35	50
	CS-515-MJP	Lab Course on CS-514-MJ	-	2	--	4	15	35	50
RM	CS-541-RM	Research Methodology	4	-	4	--	30	70	100
		Total	**16**	**6**					

Savitribai Phule Pune University

Syllabus Structure as per NEP Guidelines

M. Sc. (Computer Science) Part-I from 2023-24

SEMESTER II

Course Type	Course code	Course Name	Credits		Teaching Scheme Hrs/Week		Examination Scheme and Marks		
			TH	PR	TH	PR	CE	EE	Total
Major Core	CS-551-MJ	Design and Analysis of Algorithms	4	-	4	--	30	70	100
	CS-552-MJ	Mobile App Development Technologies	4	-	4	--	30	70	100
	CS-553-MJ	Software Project Management	2	-	2	--	15	35	50
	CS-554-MJP	Lab course on CS-551-MJ	-	2	--	4	15	35	50
	CS-555-MJP	Lab course on CS-552-MJ	-	2	--	4	15	35	50
Major Elective	CS-560-MJ	Full Stack Development - I	2	-	2	--	15	35	50
	CS-561-MJP	Lab Course on CS-560-MJ	-	2	--	4	15	35	50
	OR								
	CS-562-MJ	Web Services	2	-	2	--	15	35	50
	CS-563-MJP	Lab Course on CS-562-MJ	-	2	--	4	15	35	50
	OR								
	CS-564-MJ	ASP.NET Programming	2	-	2	--	15	35	50
	CS-565-MJP	Lab course on CS-564-MJ	-	2	--	4	15	35	50
On Job Training	CS-581-OJT	On Job Training/Internship	-	4	-	-	30	70	100
		Total	**12**	**10**					

Savitribai Phule Pune University

Syllabus Structure as per NEP Guidelines

M. Sc. (Computer Science) Part-II from 2024-25

SEMESTER III

Course Type	Course code	Course Name	Credits		Teaching Scheme Hrs/Week		Examination Scheme and Marks		
			TH	PR	TH	PR	CE	EE	Total
Major Core	CS-601-MJ	Software Architecture and Design Pattern	4	-	4	--	30	70	100
	CS-602-MJ	Machine Learning	4	-	4	--	30	70	100
	CS-603-MJ	Internet of Things	2	-	2	--	15	35	50
	CS-604-MJP	Lab course on CS-601-MJ and 603	-	2	--	4	15	35	50
	CS-605-MJP	Lab course CS-602-MJ	-	2	--	4	15	35	50
Major Elective	CS-610-MJ	Full Stack Development- II	2	-	2	--	15	35	50
	CS-611-MJP	Lab Course on CS-610-MJ	-	2	--	4	15	35	50
	OR								
	CS-612-MJ	DevOps Fundamentals	2	-	2	--	15	35	50
	CS-613-MJP	Lab Course on CS-612-MJ	-	2	--	4	15	35	50
	OR								
	CS-614-MJ	Soft Computing	2	-	2	--	15	35	50
	CS-615-MJP	Practical on CS-614-MJ	-	2	--	4	15	35	50
Research Project	CS-631-RP	Research Work - I	-	4	-	-	30	70	100
		Total	**12**	**10**					

Savitribai Phule Pune University

Syllabus Structure as per NEP Guidelines

M. Sc. (Computer Science) Part-II from 2024-25

SEMESTER IV

Course Type	Course code	Course Name	Credits		Teaching Scheme Hrs/Week		Examination Scheme and Marks		
			TH	PR	TH	PR	CE	EE	Total
Major Core	CS-651-MJP	Full Time Industrial Training (IT)	-	16	-	-	120	280	400
Research Project	CS-681-RP	Research Work -II	-	6	-	-	45	105	150
		Total	**-**	**22**					

Abbreviations

CS	Computer Science		**MJ**	Major Theory
RM	Research Methodology		**MJP**	Major Practical
OJT	On Job Training		**RP**	Research Project
TH	Theory		**PR**	Practical
CE	Continuous Evaluation		**EE**	End semester Evaluation
MOOC	Massive Open Online Course			

Savitribai Phule Pune University

M.Sc. Computer Science (From 2024-25) Sem-III

CS-601-MJ :Software Architecture and Design Pattern

No. of Credits:04	Teaching Scheme Theory: 4 Hrs/Week	Examination Scheme Continuous Evaluation:30Marks End Semester:70 Marks

Prerequisite

- Familiarity with UML and OOPs Concepts
- Programming in Java & CPP

Objectives

- To introduce students to the foundation ideas, methods and techniques of Software Architecture and Design Patterns.
- To write java programs and applications that make use of frameworks and design patterns to create reusable and flexible software systems.
- To make use of patterns and architectures for solving practical problems.
- To clear idea about various design pattern.
- To understand about the process of deploying web apps using specific Frameworks.

Course Outcomes

On Completion of this course, student will be able to -

CO1: Understand the UML basics, RUP and basics of software architecture

CO2: Acknowledge the traits of patterns that make them helpful in solving real-world issues.

CO3: Able to use specific frameworks as per applications need.

CO4: Design java application using design pattern techniques.

Unit No.	Name of Unit	Teaching Hours	CO Targeted
1	**Introduction**	**3**	**CO1**
1.1 UML The Notation 1.2 Process Unified Process / Rational Unified Process inception, elaboration, construction, transition 1.3 How various components fit in the life cycle 1.4 The artifacts at end of each process / discipline			
2	**Software Architecture**	**4**	**CO1**
2.1 Definition Software Architecture and its examples 2.2 Importance of software architecture 2.3 Architectural structures and views			
3	**Architectural Styles**	**10**	**CO2**
3.1 Architectural Styles 3.2 Pipes and Filters 3.3 Layered Systems 3.4 Data Abstraction and Object – Oriented Organization			

3.5 Repositories, Interpreters 3.6 Heterogeneous Architectures.			
4	**Introduction to Patterns**	**5**	**CO2**
4.1 Meaning of Pattern, Definition of Design Pattern 4.2 What makes a Pattern (GOF) 4.3 Describing Design Patterns. 4.4 Pattern Categories & Relationships between Patterns 4.5 Organizing the Catalogue 4.6 Patterns and Software Architecture			
5	**Study of Design Patterns**	**18**	**CO4**
5.1 Creational Patterns-singleton, factory method, abstract factory 5.2 Structural Patterns-adapter, decorator, façade 5.3 Behavioural Patterns-iterator, observer, strategy, command and state (study of intent, applicability, participants, structure, collaboration , Java Example code , Implementation and consequences			
6	**GRASP(General Responsibility Assignment Software Patterns)**	**08**	**CO4**
6.1 Expert, Creator, High Cohesion, Low Coupling 6.2 Controller, Polymorphism, Pure Fabrication, Indirection			
7	**Study of Frameworks**	**08**	**CO3**
7.1 Frameworks as reusable chunks of architecture 7.2 The framework lifecycle, development using frameworks 7.3 Spring Core Framework 7.4 Spring Boot Framework 7.5 Spring and AOP 7.6 DAO Support and JDBC Framework 7.7 Spring and EJB 7.8 Advantages of Spring			
8	**Case Study**	**04**	**CO3,CO4**
8.1 Take a Framework and find Patterns in the Framework. 8.2 Benefits of Patterns in the chosen Framework			

Reference Books

1. Design Patterns – Elements of Reusable Object-oriented Software By E. Gamma, Richard Helm, Ralph Johnson , John Vlissides (GoF)
2. Pattern – Oriented Software Architecture (POSA) Volume 1. By : Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal.
3. Software Architecture in Practice. By Len Bass, Paul Clements, Rick Kazman
4. Applying UML and Patterns By Craig Larman.
5. Software Architecture- Perspectives on an emerging discipline by Mary shaw and David Garlan
6. Head First Design Pattern by Kathy Sierra, Bert Bates, Elisabeth Robson, Eric Freeman Publisher: OReilly Media, Inc.
7. Building Microservices-Designing Fine-Grained Systems By Sam Newman Publisher: OReilly Media
8. Design patterns in Java by Douglas Schmidt Publisher O'Reilly

9. Professional Java Development with the Spring Framework 1st Edition
by Rod Johnson, Alef Arendsen, Thomas Risberg, Colin Sampaleanu ; WROX publication
10. Mastering Spring 5: An effective guide to build enterprise applications using Java Spring and Spring Boot framework, 2nd Edition by Ranga Rao Karanam ; PACKT publishing
11. The Unified Modeling Language reference manual Second Edition By James Rumbaugh, Ivar Jacobson, Grady Booch Publisher: Pearson Higher Education

Web References:

You tube Link : (MicroServices)
https://www.youtube.com/playlist?list=PLqq-6Pq4lTTZSKAFG6aCDVDP86Qx4lNas
Spring Framework –
https://youtu.be/GB8k2-Egfv0
https://www.baeldung.com/spring-vs-spring-boot
https://www.baeldung.com/spring-boot
https://www.youtube.com/watch?v=msXL2oDexqw&list=PLqq-6Pq4lTTbx8p2oCgcAQGQyqN8XeA1x

Lecture Notes

https://www.tutorialspoint.com/spring/spring_architecture.htm

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-602-MJ: Machine Learning

No. of Credits:4	Teaching Scheme Theory:4 Hrs/Week	Examination Scheme Continuous Evaluation:30 Marks End Semester:70 Marks

Prerequisite

- Familiarity with Probability Theory, Multivariable Calculus, Linear Algebra
- Programming in Python (NumPy, SciPy, Pandas, Matplotlib, Seaborn, SciKit-Learn, Statistics Model, Tensor Flow)

Objectives

- To introduce students to the basic concepts and techniques of Machine Learning.
- To write python programs using machine learning algorithms for solving practical problems.
- To understand about Machine Learning Library and use cases.
- To understand about the process of deploying ML model.

Course Outcomes

On Completion of this course, student will be able to -

CO1: To introduce knowledge of Machine Learning.
CO2: To demonstrate all categories of Machine learning algorithms along with implementation.
CO3: To compose real time application using machine learning algorithms.
CO4: Analyze the concept of neural networks for learning linear and non-linear activation functions.

Unit No.	Name of Unit	Teaching Hours	CO Targeted
1	**Introduction to Machine Learning**	**10**	**CO1**

1.1 Concepts of Data Science, Artificial Intelligence, Machine Learning and Deep Learning
1.2 Why learn and what is learning?
1.3 What is Machine Learning?
1.4 Traditional Programming Vs. Machine Learning
1.5 Machine Learning Process, Types of Data, Key Elements of Machine Learning (Representation, Evaluation and Optimization), Feature Selection techniques (filter and wrapper method).
1.6 Descriptive and Inferential Statistics: Probability, Normal Distribution, Distance Measures (Euclidean and Manhattan), Correlation Techniques, Introduction to Hypothesis Testing.
1.7 Creating our own dataset, Importing the dataset, Handling Missing Data, Cross validation methods, Feature Scaling (Normalization and standardization).
1.8 Type of Learning- Supervised, Unsupervised and Semi-Supervised Learning
1.9 Components of Generalization Error (Bias, Variance, under fitting, over fitting)

2	**Supervised Machine Learning**	**13**	**CO1, CO2, CO3**

2.1. Introduction to Supervised Machine Learning
2.2. Regression in Machine Learning:
  2.2.1. Types of Regression Algorithms-Linear Regression, Polynomial, Stepwise, Ridge, Lasso, Elastic Net, Bayesian Regression.
  2.2.2 Evaluation methods for regression: MAE, RMSE, R-Square error
2.3. Classification in ML: Classification Algorithms
  2.3.1.Logistic Regression
  2.3.2.Decision Tree
  2.3.3. k-Nearest Neighbors
  2.3.4.Naive Bayes
  2.3.5.Support Vector Machines (SVM)
  2.3.6. Evaluation methods: Confusion matrix, precision, recall, F1-score, Accuracy, ROC-AUC curve,
2.4. Non-linear regression: Decision Tree, Support Vector, KNN
2.5. Advantages and Disadvantages of supervised ML

3	**Unsupervised Machine Learning**	**11**	**CO1, CO2, CO3**

3.1. Introduction to Unsupervised Learning
3.2. Clustering Methods
  3.2.1. Density-Based Methods
  3.2.2. Hierarchical-Based Methods
  3.2.3. Partitioning-Based Methods
  3.2.4. Grid-Based Methods
3.3. Clustering Algorithms
  3.3.1 K-means clustering
  3.3.2 Hierarchical Clustering (Agglomerative, Divisive), Dendrogram
  3.3.3 Selecting optimal number of clusters: Within Clusters Sum of Squares (WCSS) by Elbow Method, Silhouette Score
3.4. Introduction to Association rules
3.5. Association Algorithms- Apriori Algorithm
3.6. Dimensionality Reduction – Singular value decomposition, Principle Component Analysis(PCA)
3.7. Anomaly detection (outlier detection)
3.8. Independent Component Analysis
3.9. Advantages and Disadvantages of Unsupervised ML

4	**Ensemble Learning**	**5**	**CO1, CO2, CO3**

4.1.Model Combination Schemes
4.2.Bagging: Random Forest Trees
4.3.Boosting: Adaboost, Gradient boost
4.4.Voting
4.5. Stacking
4.6.Error-Correcting Output Codes
4.7.Gaussian mixture models
4.8.The Expectation-Maximization (EM) Algorithm

5	**Reinforcement Learning**	**6**	**CO1, CO2, CO3**
5.1 Upper Confidence Bound 5.2 Thompson Sampling 5.3 Q-Learning 5.4 Applications of reinforcement learning			
6	**Neural Network And Deep Learning**	**15**	**CO1, CO2, CO3, CO4**
6.1 Introduction to Deep Learning, Deep learning Vs Machine learning 6.2 Types of Neural Network- Artificial Neural Network (ANN), Convolution Neural Network (CNN), Recurrent Neural Network (RNN) 6.3 Introduction Neural Network 6.3.1. Perceptron/ Delta Learning Rule 6.3.2. Activation functions 6.3.3.Role of Loss Functions and Optimization 6.3.4. Gradient Descent 6.3.5. Multi-Layer Perceptron (MLP) 6.3.6. Backpropagation 6.3.7. MLP for Classification and Regression 6.3.8. Regularization 6.3.9.Early Stopping 6.3.10. Batch Normalization 6.4. Introduction to Convolution Neural Network(CNN) 6.5. Introduction to Recurrent Neural Network (RNN)			

Reference Books

1. Mitchell, Tom M."Machinelearning.WCB."(1997).
2. Alpaydin,E.2010. Introduction to Machine Learning. 2nd edition, MIT.
3. Ethem Alpaydin: Introduction to Machine Learning, PHI 2nd Edition-2013.
4. Andreas C. Müller & Sarah Guido: Introduction to Machine Learning with Python A Guide for Data Scientists- O'Reilly publications
5. Rogers, Simon, and Mark Girolami. A first course in machine learning.CRCPress,2015.
6. Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning.Vol.1. Springer, Berlin: Springer series in statistics, 2001.
7. Witten, Ian H., and Eibe Frank. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005.
8. Machine learning course material by Andrew Ng, Stanford university
9. Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: Anintroduction. Vol.1. No.1. Cambridge: MIT press,1998.
10. Iba, Takashi, etal. "Learning patterns: A pattern language for active learners. "Conference on Pattern Languages of Programs (PLoP). 2009.
11. Machine Learning in Data Science Using Python by Dr. R. Nageswara Rao Dreamtech press, 2023.
12. https://machinelearningmastery.com/

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-603-MJ :Internet of Things

No. of Credits:02	Teaching Scheme Theory : 2 Hrs/Week	Examination Scheme Continuous Evaluation: 15 Marks End Semester: 35 Marks

Prerequisite

- Students Should have basic knowledge of Networking, Internet and Electronics
- Programming Skills: Proficiency in programming languages, particularly C/C++ and Python.

Objectives

- To understand the fundamentals of Internet of Thing
- To build a small low cost embedded system using Arduino / Raspberry Pi or equivalent Boards
- To create a network of connected devices.
- To apply the concept of Internet of Things in the real world scenario.

Course Outcomes

On Completion of this course, student will be able to -

CO1: Demonstrate basic concepts, principles and challenges in IoT.

CO2: Illustrate functioning of hardware devices and sensors used for IoT.

CO3: Analyze network communication aspects and protocols used in IoT.

CO4: Apply IoT for developing real life applications using Ardunio programming.

CO5: To develop IoT infrastructure for popular applications.

Unit No.	Name of Unit	Teaching Hours	CO Targeted
1	**Fundamentals of IoT**	**7**	**CO1**

1.1 Concepts and Definitions of The Internet of Things (IoT),History of IoT

1.2 Characteristics, Conceptual Framework, Architectural view, technology behind IoT, source of the IoT (Zetta), IoT Examples.

1.3 Design Principles for Connected Devices: IoT/M2M systems layers and design standardization, Physical vs. logical design, communication technologies, data enrichment and consolidation, ease of designing and afford ability.

1.4 Major components of IoT devices (sensors or Gateway, cloud, Analytics, User Interface).

2	**Hardware for IoT**	**6**	**CO2**

2.1 Sensors, Digital sensors, actuators, wireless sensor networks, participatory sensing technology.

2.2 IOT Protocol: MQTT, CoAP, XMPP.

2.3 Embedded Platforms for IoT: Embedded computing basics(block diagram), Overview of IOT supported Hardware platforms such as Arduino, Raspberry pi.

3	**Network and Communication aspects in IoT**	**7**	**CO3**

3.1 Wireless Medium access issues, MAC protocol survey, Survey routing protocols, Sensor

deployment & Node discovery, Data aggregation & Dissemination

3.2 Communication Technologies and Features: Bluetooth, ZigBee, Zwave, RFID, GPS, NFC, Ethernet TCP/IP.

3.3 Cloud based Architecture, SaaS, PaaS and IaaS, Benefits risk and challenges of cloud computing platforms and services, Introduction to cloud based IoT Platforms like IBM, Thingspeak, AWS etc.

4	**Programming for IoT**	6	CO4

4.1 Arduino Software Setup the IDE, Writing Arduino Software, The Arduino Sketch, Some Basic Examples, Trying the code on an Arduino Emulator.

4.2 Arduino Libraries, Programming & Interfacing, Programming Arduino for the IoT- Using Timers, Threads, Adding Security to Sensor Readings, Authenticating and Encrypting Arduino Data.

4.3 Introduction to Raspberry PI, Installation, GPIO, Interfacing, Programming, Features of Python.

5	**IoT Application and Case study**	4	CO5

5.1 Development Challenges, Security Challenges, Smart Metering, E-health, City Automation

5.2 Automotive Applications, home automation, smart cards, communicating data with H/W. units, mobiles, tablets, Designing of smart street lights in smart city.

Reference Books

1.Olivier Hersent, David Boswarthick, Omar Elloumi "The Internet of Things key applications and protocols", willey

2. Jeeva Jose, Internet of Things, Khanna Publishing House

3. Michael Miller "The Internet of Things" by Pearson

4. Raj Kamal "INTERNET OF THINGS", McGraw-Hill, 1ST Edition, 2016

5. Arshdeep Bahga, Vijay Madisetti "Internet of Things (A hands on approach)" 1ST edition, VPI publications,2014

6. Adrian McEwen, Hakin Cassimally "Designing the Internet of Things" Wiley India

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-604-MJP :Lab Course on CS-601-MJ and CS-603-MJ (Software Architecture & Design Pattern and Internet of Things)

No. of Credits: 02	**Teaching Scheme** Practical: 04 Hrs/Week	**Examination Scheme** Continuous Evaluation:15 Marks End Semester: 35 Marks

Prerequisite

- Familiarity with UML and OOPs Concepts
- Programming in Java & CPP
- Basic knowledge of Networking, Internet and Electronics
- Programming Skills: Proficiency in programming languages, particularly C/C++ and Python.

Objectives

- To write java programs and applications that make use of frameworks and design patterns to create reusable and flexible software systems.
- The student should have hands on experience in using various sensors like temperature, humidity, smoke, light, etc. and should be able to use control web camera, network and relays connected to the Pi.

Course Outcomes

On Completion of this course, student will be able to -

CO1: Design java application using design pattern techniques.

CO2: Apply IoT for developing real life applications using Ardunio programming.

CO3: To develop IoT infrastructure for popular applications.

Assignment based on Software Architecture and Design Patterns (SADP)

1. Write a JAVA Program to implement built-in support (java.util.Observable) Weather station with members temperature, humidity, pressure and methods - mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(), getPressure()
2. Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.
3. Write a Java Program to implement Factory method for Pizza Store with createPizza(), orderPizza(),prepare(), bake(), cut(), box(). Use this to create variety of pizza's like NyStyleCheesePizza, ChicagoStyleCheesePizzaetc.
4. Write a Java Program to implement Singleton pattern for multithreading.
5. Write a Java Program to implement command pattern to test Remote Control. Book
6. Write a Java Program to implement undo command to test Ceilingfan.
7. Write a Java Program to implement Iterator Pattern forDesigning Menu like Breakfast, Lunch or DinnerMenu.

Case Study (to be performed using pen and paper and submitted as assignment)

1.Case study on any 4 UMLdiagrams of any system.

2.Draw UML diagrams using softwares (such as Rational Rhapsody or Eclipse UML2 tools etc.)

Assignments based on Internet of Things (IoT)

1. To write a program to sense the available networks using Arduino
2. To write a program to measure the distance using ultrasonic sensor and make LED blink using Arduino.
3. To write a program to detects the vibration of an object with sensor using Arduino.
4. To write a program to sense a finger when it is placed on the board Arduino.
5. To write a program to connect with the available Wi-Fi using Arduino.
6. To write a program to get temperature notification using Arduino.
7. To write a program for LDR to vary the light intensity of LED using Arduino.
8. Start Raspberry Pi and try various Linux commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, pingetc.
9. Run some python programs on Pi like:
   a) Read your name and print Hello message with name
   b) Read two numbers and print their sum, difference, product and division.
   c) Word and character count of a given string.
   d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.
10. Run some python programs on Pi like:
    a) Print a name 'n' times, where name and n are read from standard input, using for and while loops.
    b) Handle Divided by Zero Exception.
    c) Print current time for 10 times with an interval of10seconds.
    d) Read a fileline byline and print the word count of each line
11. Run some python programs on Pi like
    a) Light an LED through Python program
    b) Get input from two switches and switch on corresponding LEDs
    c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-605-MJP:Lab course on CS-602-MJ (Machine Learning)

No. of Credits:2	Teaching Scheme Practical: 4 Hrs/Week	Examination Scheme Continuous Evaluation:15 Marks End Semester: 35 Marks

Prerequisite

- Programming in Python
- Data preprocessing methods
- Statistical methods (mean, median, mode etc.…)
- Data Visualization methods, tools, types

Objectives

- To implement machine learning algorithms for solving real life problems.
- To understand about the methods of deploying ML model.
- To understand Machine learning algorithm for predicting outcomes.
- Apply appropriate Machine Learning Methods and tools to analyze data, create models, and identify insights that can lead to actionable results.
- To Implement neural networks (ANN, RNN and CNN)

Course Outcomes

On Completion of this course, student will be able to -

CO1: To Get Hands on machine learning model.

CO2: Able to estimate Machine Learning models efficiency using suitable metrics.

CO3: Able to analysis and make decision for critical problems.

CO4: Able to handle structured, unstructured as well as semi-structured data.

CO5: Implement ideas to design and develop Deep learning solutions for complex problems. .

Assign No.	Assignment
1	Create a multiple linear regression model for house price dataset divide dataset into train and test data while giving it to model and predict prices of house.
2	Use dataset crash.csv is an accident survivor's dataset portal for USA hosted by data.gov. The dataset contains passengers age and speed of vehicle (mph) at the time of impact and fate of passengers (1 for survived and 0 for not survived) after a crash. use logistic regression to decide if the age and speed can predict the survivability of the passengers.
3	Fit the simple linear regression and polynomial linear regression models to Salary_positions.csv data. Find which one is more accurately fitting to the given data. Also predict the salaries of level 11 and level 12 employees.
4	Write a python program to categorize the given news text into one of the available 20 categories of news groups, using multinomial Naïve Bayes machine learning model.
5	Implement Ridge Regression, Lasso regression, ElasticNet model using

	boston_houses.csv and take only 'RM' and 'Price' of the houses. divide the data as training and testing data. Fit line using Ridge regression and to find price of a house if it contains 5 rooms. and compare results.
6	Write a python program to Implement Decision Tree classifier model onData which is extracted from images that were taken from genuine and forged banknote-like specimens. (refer UCI dataset https://archive.ics.uci.edu/dataset/267/banknote+authentication)
7	Classify the iris flowers dataset using SVM and find out the flower type depending on the given input data like sepal length, sepal width, petal length and petal width Find accuracy of all SVM kernels.
8	Create KNN model on Indian diabetes patient's database and predict whether a new patient is diabetic (1) or not (0). Find optimal value of K.
9	Implement Non-linear regression model (Decision Tree,SVM,KNN)to predict the consumption of petrol use petrolconsumption dataset.(https://www.kaggle.com/code/ajinkyaa/linear-regression-petrol-consumption)
10	Take iris flower dataset and reduce 4D data to 2D data using PCA. Then train the model and predict new flower with given measurements.
11	Use K-means clustering model and classify the employees into various income groups or clusters. Preprocess data if require (i.e. drop missing or null values). Use elbow method and Silhouette Score to find value of k.
12	The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units on diverse product categories. Using data Wholesale customer dataset compute agglomerative clustering to find out annual spending clients in the same region. **https://archive.ics.uci.edu/dataset/292/wholesale+customers**
13	Use Apriori algorithm on groceries dataset to find which items are brought together. Use minimum support =0.25
14	Implement Ensemble ML algorithm on Pima Indians Diabetes Database with bagging (random forest), boosting, voting and Stacking methods and display analysis accordingly. Compare result.
15	Create a two layered neural network with relu and sigmoid activation function.
16	Create an ANN and train it on house price dataset classify the house price is above average or below average.
17	Create a CNN model and train it on mnist handwritten digit dataset. Using model find out the digit written by a hand in a given image. **Import mnist dataset from tensorflow.keras.datasets**
18	Create RNN model and analyze the Google stock price dataset. Find out increasing or decreasing trends of stock price for the next day.
	ML Tool: Power BI public **Reference datasets:** https://www.kaggle.com/datasets/manmohan291/housepricedata.csv https://www.kaggle.com/uciml/iris https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset/input

	https://www.kaggle.com/datasets/irfanasrullah/groceries https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset https://www.kaggle.com/datasets/gurdit559/canada-per-capita-income-single-variable-data-set https://www.kaggle.com/datasets/vaibhavsxn/google-stock-prices-training-and-test-data https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents https://archive.ics.uci.edu/datasets

Savitribai Phule Pune University
M.Sc. Computer Science (2024-25) Sem-III

CS-610-MJ : Full Stack Development-II

No. of Credits: 2	Teaching Scheme Theory:2 Hrs/Week	Examination Scheme Continuous Evaluation:15Marks End Semester:35Marks

Prerequisite

- Knowledge of HTML, CSS, JavaScript, Angular Framework, ES6, TypeScript, NodeJS, ExpressJS and MongoDB

Objectives

- Get deep understanding of Angular framework and RxJS library.
- Learn State management with NgRx.
- Create dynamic components with custom life-cycle hooks.
- Build scalable and reusable features.
- Learn advanced Typescript concepts.
- Build a more reliable Angular app with type safety.
- Learn Node JS In depth.
- Build industry grade Restful APIs with ExpressJS.
- Learn advanced MongoDB concepts.
- Learn Best practices related to application performance optimization, security, testing.

Course Outcomes

On Completion of this course, student will be able to -

CO1: Learn In Depth understanding of Angular framework and State Management.

CO2: Learn using typescript effectively in Angular framework.

CO3: Learn in-depth knowledge of NodeJS and Express JS.

CO4: Learn advance concepts in MongoDB.

CO5: Learn best practices to be followed when creating industry grade applications.

Unit No.	Name of Unit	Teaching Hours	CO Targeted
1	**Deep Dive into Angular Framework**	**12**	**CO1**

1.1 Recap of Angular fundamentals (components, directives, services, dependency injection)

1.2 Introduction to advanced topics covered in the course

1.3 Prerequisite knowledge refresher in ES6, TypeScript, Node.js, Express.js, and MongoDB.

1.4 Introduction to RxJS library

- 1.4.1 Observable
- 1.4.2 Observer
- 1.4.3 Subscription
- 1.4.4 Operators
- 1.4.5 Subject
- 1.4.6 Schedulers

1.5 State management with NgRx

1.5.1 @ngrx/store
1.5.2 @ngrx/effects
1.5.3 @ngrx/router-store
1.5.4 @ngrx/entity
1.5.5 @ngrx/component-store
1.5.6 @ngrx/signals
1.5.7 @ngrx/operators
1.5.8 @ngrx/data
1.5.9 @ngrx/component
1.5.10 Developer Tools
1.5.10.1 @ngrx/store-devtools
1.5.10.1 @ngrx/schematics
1.5.10.1 @ngrx/eslint-plugin
1.6 Custom life-cycle hooks
1.7 Communication between components beyond @Input() and @Output().
1.8 Advanced routing concepts (lazy loading, nested routes, route guards, resolvers)
1.9 Building Scalable and Reusable Features
1.9.1 Feature modules and lazy loading for modularity
1.9.2 Creating custom directives and pipes
1.9.3 Services for data fetching and business logic
1.9.4 Dependency injection best practices

2	**Mastering Type Script**	4	CO2

2.1. Advanced TypeScript Features
2.1.1 Generics for type safety and code reusability
2.1.2 Decorators for custom functionality and metadata
2.1.3 Interfaces and advanced type annotations
2.1.4 Utility types (mapped types, conditional types)
2.2. Building a Type-Safe Angular Application
2.2.1 Leveraging TypeScript to enforce data types in components, services, and other parts of application.
2.2.2 Using interfaces to define contracts
2.2.3 Writing unit tests with type safety

3	**Mastering Node.js and Express.js**	4	CO3

3.1 Node.js in Depth
3.1.1 Asynchronous programming with promises and async/await
3.1.2 Event loop and non-blocking I/O
3.1.3 Modules and the Node.js package manager (npm)
3.2 Building a RESTful API with Express.js
3.2.1 Creating routes for handling HTTP requests
3.2.2 Middleware for request processing and error handling
3.2.3 Body parsing and validation
3.2.4 Sending JSON responses

4	**MongoDB for Data Persistence**	4	CO4

4.1 Advanced Mongoose Features
4.1.1 Schema validation and middleware

4.1.2 Population, aggregation pipelines, and custom queries
4.1.3 Mongoose with TypeScript for type safety
4.2 Database Authentication and Security
4.2.1 User authentication and authorization
4.2.2 Data encryption at rest and in transit
4.2.3 Best practices for secure MongoDB deployments
4.3 Scalability and High Availability
4.3.1 Sharding for horizontal scaling
4.3.2 Replication for data redundancy and failover
4.3.3 MongoDB Atlas for managed database services

5	**Advanced Topics and Best Practices**	6	**CO5**

5.1 Performance Optimization Techniques
5.1.1 Code splitting and lazy loading for faster initial load times
5.1.2 Change detection strategies (OnPush)
5.1.3 Caching mechanisms (local storage, service workers)
5.2 Security Considerations
5.2.1 Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) prevention
5.2.2 Input validation and sanitization
5.2.3 Secure communication with the backend (HTTPS)
5.3 Testing Strategies
5.3.1 Unit testing with Jest or Karma
5.3.2 End-to-end testing with Cypress or Playwright
5.3.3 Best practices for writing effective tests

Reference Books

1. RxJS in action by Paul P. Daniels and Luis Atencio
2. Basics Of Javascript Rxjs By Antonette Milby
3. Reactive Programming with Angular and ngrx By Oren Farhi
4. Reactive Patterns with RxJS and Angular Signals By Lamis Chebbi 2024 edition
5. Reactive State for Angular with NgRx by Amit Gharat
6. Mastering Angular 16: Advanced Techniques for Web Development By Raman Kumar
7. Angular Enterprise Architecture: by Tomas Trajan
8. Angular Development with TypeScript 2nd edition by Yakov Fain and Anton Moiseev
9. Mastering TypeScript - Fourth Edition By Nathan Rozentals
10. TypeScript Deep Dive By Basarat Ali Syed
11. Programming TypeScript: Making Your JavaScript Applications Scale By Boris Cherny
12. Essential TypeScript: From Beginner to Pro By Adam Freeman
13. Hands-On Functional Programming with TypeScript By Remo H. Jansen
14. Mastering Node.js – Second Edition By Sandro Pasquali and Kevin Faaborg
15. Advanced Node.js Development by Andrew Mead
16. Node.js Design Patterns – Second Edition By Mario Casciar
17. Express in Action. Writing, Building, and Testing Node.js Applications By Evan M. Hahn
18. Express.js Deep API Reference by Azat Mardan
19. Mastering MongoDB 7.0 - 4th Edition By Marko Aleksendrić , Arek Borucki , Leandro Domingues

20. Clean JavaScript A concise guide to learning Clean Code, SOLID and Unit Testing By Miguel A. Gómez Álvarez

Web Links

1. https://rxjs.dev/guide/overview
2. https://ngrx.io/docs
3. https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
4. https://expressjs.com/
5. https://learn.mongodb.com/
6. https://angular.dev/overview
7. https://angular.dev/guide/testing
8. https://www.w3schools.com/
9. https://www.tutorialspoint.com/
10. https://www.geeksforgeeks.org/
11. https://www.freecodecamp.org/
12. https://www.coursera.org/
13. https://www.udemy.com/
14. https://www.edx.org/
15. https://alison.com/courses

Savitribai Phule Pune University

M.Sc. Computer Science (From 2024-25) Sem-III

CS-611-MJP:Lab course on CS-610-MJ (Full Stack Development-II)

No. of Credits:2	Teaching Scheme Practical: 4 Hrs/Week	Examination Scheme Continuous Evaluation:15 Marks End Semester: 35 Marks

Prerequisite

- Knowledge of HTML, CSS, JavaScript, Angular Framework, ES6, TypeScript, NodeJS, ExpressJS and MongoDB

Objectives

- Get deep understanding of Angular framework and RxJS library.
- Learn State management with NgRx.
- Create dynamic components with custom life-cycle hooks.
- Build scalable and reusable features.
- Learn advanced Typescript concepts.
- Build a more reliable Angular app with type safety.
- Learn Node JS In depth.
- Build industry grade Restful APIs with ExpressJS.
- Learn advanced MongoDB concepts.
- Learn Best practices related to application performance optimization, security, testing.

Course Outcomes

On Completion of this course, student will be able to -

CO1: Learn In Depth understanding of Angular framework and State Management.

CO2: Learn using typescript effectively in Angular framework.

CO3: Learn in-depth knowledge of NodeJS and Express JS.

CO4: Learn advance concepts in MongoDB.

CO5: Learn best practices to be followed when creating industry grade applications.

Assign No.	Assignments
1	Create a simple Angular application that fetches data from an API using HttpClient. - Implement an Observable to fetch data from an API endpoint. - Use operators like `map`, `filter`, `tap`, etc., to manipulate the fetched data. - Display the fetched data in the Angular application using `async` pipe.
2	Build a simple TODO application with NgRx for state management. - Define the actions, reducers, selectors, and effects for managing TODOs. - Implement CRUD operations (Create, Read, Update, Delete) for TODOs. - Display TODOs in the UI and allow users to add, edit, and delete them.
3	Create a custom life-cycle hook for logging component life-cycle events - Define a custom life-cycle hook named `LogLifecycle` that logs component initialization, destruction, and changes in input properties. - Implement the custom hook in a sample component.

	- Test the behavior by adding the component to a parent component and observing the logs.
4	Implement a service for communication between sibling components - Create a service named `DataService` with methods to send and receive data. - Use this service to enable communication between two sibling components. - Demonstrate passing data from one component to another without using @Input() or @Output().
5	Understand how generics can enhance type safety and code reusability in TypeScript -Create a TypeScript function that accepts an array of any type and returns the first element of that array. - Refactor the function to use generics to enforce type safety. - Create a generic class that implements a basic stack data structure (push, pop, peek). - Test the stack class with different data types.
6	Explore the use of decorators for adding custom functionality and metadata to TypeScript classes. - Create a custom decorator `@Log` that logs method calls with their arguments. -Apply the `@Log` decorator to a few methods in a class and observe the logged output. -Implement a validation decorator `@Validate` that ensures method arguments meet certain criteria. - Apply the `@Validate` decorator to methods with different validation rules.
7	Apply TypeScript to enforce data types and enhance type safety in an Angular application - Create a simple Angular component that takes input data and displays it. - Use TypeScript interfaces to define the structure of input data passed to the component. - Implement type-safe services by defining interfaces for API responses and request payloads. - Write unit tests for the component and services using TypeScript to ensure type safety. - Explore Angular features like Dependency Injection and HttpClient with type safety in mind.
8	Create a Node.js application that reads data from multiple files asynchronously using promises and async/await.
9	Implement a simple server using Node.js that handles multiple client connections concurrently.
10	Build a small application that uses multiple modules to perform specific tasks (e.g., file manipulation, data processing).
11	Develop an Express.js application that defines routes for CRUD operations (Create, Read, Update, Delete) on a resource (e.g., users, products).
12	Implement middleware functions in an Express.js application to log incoming requests and handle errors gracefully.
13	Extend the previous Express.js application to include middleware for parsing request bodies (e.g., JSON, form data) and validating input data. Send appropriate JSON responses for success and error cases.

14	Create a Node.js application that uses Mongoose to define a schema for a user object with validation rules (e.g., required fields, data types). Implement Mongoose middleware functions to perform pre and post document operations (e.g., hashing passwords before saving).
15	Extend the previous application to include relationships between different entities (e.g., users and posts). Use Mongoose population to fetch related documents. Implement custom queries using Mongoose aggregation pipelines to perform complex data transformations.
16	Develop a Node.js application that implements user authentication and authorization using Mongoose and a popular authentication middleware (e.g., Passport.js). Define user roles and permissions to restrict access to certain resources.
17	Enhance the previous application to encrypt sensitive user data (e.g., passwords) at rest using encryption libraries (e.g., bcrypt) and ensure data is transmitted securely over HTTPS.

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-612-MJ : DevOps Fundamentals

No. of Credits: 2	Teaching Scheme Theory: 2 Hrs/Week	Examination Scheme Continuous Evaluation: 15 Marks End Semester: 35Marks

Prerequisite

- Understanding of Software Development: Basic knowledge of programming languages, version control systems, and software development methodologies.
- Familiarity with System Administration: Fundamental understanding of operating systems, networks, and server management.
- Scripting and Automation Skills: Proficiency in scripting languages (e.g., Bash, Python) and an appreciation for automation principles to streamline processes.
- Communication and Collaboration: Strong communication skills and an ability to collaborate effectively with cross-functional teams, promoting a DevOps culture of collaboration and shared responsibility.

Objectives

- To describe the evolution of technology & timeline (Understand)
- To understand various Devops platforms (Remember)
- To demonstrate the building components / blocks of Devops and gain an insight of the Devops Architecture. (Understand)
- To apply the knowledge gain about Devops approach across various domains (Apply)
- To build DevOps application (Apply)

Course Outcomes

On Completion of this course, student will be able to -

CO1: Apply DevOps principles for collaboration, automation, and continuous improvement.
CO2: Master version control (e.g., Git) and implement effective branching strategies.
CO3: Design and optimize CI/CD pipelines for automated and streamlined software delivery.
CO4: Utilize containerization (e.g., Docker) and orchestration tools (e.g., Kubernetes) for scalable deployments.
CO5: Implement monitoring, logging, and security practices throughout the DevOps lifecycle.
CO6: Foster effective collaboration through tools like ChatOps within cross-functional teams.
CO7: Develop skills in incident response, troubleshooting, and problem resolution.

Unit No.	Name of Unit	Teaching Hours	CO Targeted
1	**Introduction to DevOps**	**7**	**CO1,CO6**

1.1. Define Devops
1.2. What is Devops
1.3. SDLC models, Lean, ITIL, Agile
1.4. Why Devops?
1.5. History of Devops
1.6. Devops Stakeholders
1.7. Devops Goals
1.8. Important terminology
1.9. Devops perspective
1.10. DevOps and Agile
1.11. DevOps Tools
1.12. Configuration management
1.13. Continuous Integration and Deployment
1.14. Linux OS Introduction
1.15. Importance of Linux in DevOps
1.16. Linux Basic Command Utilities
1.17. Linux Administration
1.18. Environment Variables
1.19. Networking
1.20. Linux Server Installation
1.21. RPM and YUM Installation
1.22 ChatOps and collaboration tools

2	**Version Control-GIT**	**6**	**CO2**

2.1. Introduction to GIT
2.2. What is Git
2.3. About Version Control System and Types
2.4. Difference between CVCS and DVCS
2.5. A short history of GIT
2.6. GIT Basics
2.7. GIT Command Line
2.8. Installing Git
2.9. Installing on Linux
2.10. Installing on Windows
2.11. Initial setup
2.12. Git Essentials
2.13. Creating repository
2.14. Cloning, check-in and committing
2.15. Fetch pull and remote
2.16. Branching
2.17. Creating the Branches, switching the branches, merging & The branches

3	**Chef for configuration management**	**7**	**CO3**

3.1. Overview of Chef; Common Chef Terminology (Server, Workstation, Client, Repository Etc.) Servers and Nodes Chef Configuration Concepts.
3.2. Workstation Setup: How to configure knife Execute some commands to test connection between knife
and workstation.
3.3. Organization Setup: Create organization; Add yourself and node to organization.
3.4. Test Node Setup: Create a server and add to organization, check node details using knife.
3.5. Node Objects and Search: How to Add Run list to Node Check node Details.
3.6. Environments: How to create Environments, Add servers to environments.
3.7. Roles: Create roles, Add Roles to organization.
3.8. Attributes: Understanding of Attributes, Creating Custom Attributes, Defining in Cookbooks.
3.9. Data bags: Understanding the data bags, Creating and managing the Data bags, Creating the data bags using CLI and Chef Console, Sample Data bags for Creating Users.

4	**Build Tool-Maven**	4	CO4

4.1. Maven Installation
4.2. Maven Build requirements
4.3. Maven POM Builds (pom.xml)
4.4. Maven Build Life Cycle
4.5. Maven Local Repository (.m2)
4.6. Maven Global Repository
4.7. Group ID, Artifact ID, Snapshot
4.8. Maven Dependencies
4.9. Maven Plugins

5	**Docker– Containers**	**6**	**CO5**

5.1. Introduction: What is a Docker, Use case of Docker, Platforms for Docker, Dockers vs. Virtualization
5.2. Architecture: Docker Architecture., Understanding the Docker components
5.3. Installation: Installing Docker on Linux. Understanding Installation of Docker on windows.
Some Docker commands. Provisioning.
5.4. Docker Hub.: Downloading Docker images. Uploading the images in Docker Registry and AWS ECS, Understanding the containers, Running commands in container. Running multiple containers.
5.5. Custom images: Creating a custom image. Running a container from the custom image. ublishing the custom image.
5.6. Docker Networking: Accessing containers, linking containers, Exposing container ports, Container Routing.
5.7 Container orchestration with Kubernetes
5.8 Container registries , Managing containerized applications

Reference Books:

1. DevOps for Developers: Michael Hüttermann
2. DevOps: A Software Architect's Perspective: Ingo M. Weber, Len Bass, and Liming Zhu
3. Building a DevOps Culture: Jennifer Davis, Katherine Daniels. Publisher: O'Reilly
4. Practical DevOps: JoakimVeronal

5. DevOps for Dummies: Gene Kim, Kevin Behr, George, Publisher: John Wiley & Sons

Web Reference:

1. https://devops.com/
2. https://devopsinstitute.com/
3. https://aws.amazon.com/devops/
4. https://www.guru99.com/devops-tutorial.html
5. https://www.edureka.co/blog/maven-tutorial/
6. https://www.chef.io/configuration-management/

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-613-MJP : Lab Course on CS-612-MJ (DevOps Fundamentals)

No. of Credits: 2	Teaching Scheme Practical:4 Hrs/Week	Examination Scheme Continuous Evaluation: 15 Marks End Semester : 35 Marks

Prerequisite

1. Basic knowledge of linux operating systems, networks, and server management
2. Knowledge of Linux command will be an added advantage

Objectives

1. To familiarize participants with essential Git concepts and commands, enabling them to effectively use Git for version control and collaboration.
2. To demonstrate proficiency in Git, including repository setup, branching strategies, and collaborative workflows.
3. To configure a CI pipeline for automated builds, testing, and notifications, ensuring code quality and early issue detection.
4. To implement a CD pipeline for automated deployment, staging, and production release, with rollback mechanisms for reliability.

Course Outcomes

On Completion of this course, student will be able to -

CO1: Demonstrate the ability to practically implement DevOps principles through hands-on assignments in version control, CI/CD, IaC, and containerization

CO2: Develop problem-solving skills by resolving simulated incidents, enhancing the understanding of incident response and troubleshooting procedures.

CO3: Attain a comprehensive skill set covering automation, scripting, collaboration tools, and cultural transformation

CO4: Empowering participants to contribute to a collaborative and efficient DevOps culture.

Assig No.	Name of Practical Assignment
1	**Exploring Git Commands through Collaborative Coding.** **Task 0 : Install Git** **Task1 : Setting Up Git Repository** ● Open the command-line interface on your computer. ● Navigate to the directory where you want to create your Git repository. ● Run the basic git commands: git init - This initialises a new Git repository in the current directory. **Task2 : Creating and Committing Changes** ● Create a new text file named "example.txt" using any text editor. ● Add some content to the "example.txt" file. ● In the command-line interface, run the following commands:

	git status - This command shows the status of your working directory, highlighting untracked files. git add example.txt - This stages the changes of the "example.txt" file for commit. git commit -m "Add content to example.txt" - This commits the staged changes with a descriptive message. **Task3 : Exploring History** Modify the content of "example.txt." Run the following commands: git status - Notice the modified file is shown as "modified." git diff - This displays the differences between the working directory and the last commit. git log - This displays a chronological history of commits. **Task4 : Branching and Merging** Create a new branch named "feature" and switch to it: git branch feature, git checkout feature or shorthand: git checkout -b feature ● Make changes to the "example.txt" file in the "feature" branch. ● Commit the changes in the "feature" branch. ● Switch back to the "master" branch: git checkout master ● Merge the changes from the "feature" branch into the "master" branch: git merge feature **Task5 : Collaborating with Remote Repositories** ● Create an account on a Git hosting service like GitHub (https://github.com/). ● Create a new repository on GitHub. ● Link your local repository to the remote repository: git remote add origin <repository_url> ● Push your local commits to the remote repository: git push origin master
2	**Implement GitHub Operations using Git.** **Task 1: Cloning a Repository** ● Sign in to your GitHub account. ● Find a repository to clone (you can use a repository of your own or any public repository). ● Click the "Code" button and copy the repository URL. ● Open your terminal or command prompt. ● Navigate to the directory where you want to clone the repository. ● Run the following command: git clone <repository_url> ● Replace <repository_url> with the URL you copied from GitHub. ● This will clone the repository to your local machine. **Task 2: Making Changes and Creating a Branch** Navigate into the cloned repository: cd <repository_name> ● Create a new text file named "amit.txt" using a text editor. ● Add some content to the "amit.txt" file. ● Save the file and return to the command line. ● Check the status of the repository: git status

	● Stage the changes for commit: git add amit.txt ● Commit the changes with a descriptive message: git commit -m "Add content to amit.txt" ● Create a new branch named "feature": git branch feature ● Switch to the "feature" branch: git checkout feature **Task 3: Pushing Changes to GitHub** ● Add Repository URL in a variable git remote add origin <repository_url> ● Replace <repository_url> with the URL you copied from GitHub. ● Push the "feature" branch to GitHub: git push origin feature ● Check your GitHub repository to confirm that the new branch "feature" is available. **Task 4: Collaborating through Pull Requests** ● Create a pull request on GitHub: ● Go to the repository on GitHub. ● Click on "Pull Requests" and then "New Pull Request." ● Choose the base branch (usually "main" or "master") and the compare branch ("feature"). ● Review the changes and click "Create Pull Request." ● Review and merge the pull request: ● Add a title and description for the pull request. ● Assign reviewers if needed. ● Once the pull request is approved, merge it into the base branch. **Task 5: Syncing Changes** ● After the pull request is merged, update your local repository: git checkout main. git pull origin main
3	**Implement GitLab Operations using Git.** **Task 1: Creating a Repository** ● Sign in to your GitLab account. ● Click the "New" button to create a new project. ● Choose a project name, visibility level (public, private), and other settings. ● Click "Create project." **Task 2: Cloning a Repository** ● Open your terminal or command prompt. ● Navigate to the directory where you want to clone the repository. ● Copy the repository URL from GitLab. ● Run the following command: git clone <repository_url> ● Replace <repository_url> with the URL you copied from GitLab. ● This will clone the repository to your local machine. **Task 3: Making Changes and Creating a Branch** ● Navigate into the cloned repository: cd <repository_name> ● Create a new text file named "example.txt" using a text editor. ● Add some content to the "example.txt" file. ● Save the file and return to the command line. ● Check the status of the repository: git status

	● Stage the changes for commit: git add example.txt ● Commit the changes with a descriptive message: git commit -m "Add content to example.txt" ● Create a new branch named "feature": git branch feature ● Switch to the "feature" branch: git checkout feature **Task 4: Pushing Changes to GitLab** ● Add Repository URL in a variable git remote add origin <repository_url> ● Replace <repository_url> with the URL you copied from GitLab. ● Push the "feature" branch to GitLab: git push origin feature ● Check your GitLab repository to confirm that the new branch "feature" is available. **Task 5: Collaborating through Merge Requests** **1. Create a merge request on GitLab:** • Go to the repository on GitLab. • Click on "Merge Requests" and then "New Merge Request." • Choose the source branch ("feature") and the target branch ("main" or master"). • Review the changes and click "Submit merge request." **2. Review and merge the merge request:** • Add a title and description for the merge request. • Assign reviewers if needed. • Once the merge request is approved, merge it into the target branch. **Task 6: Syncing Changes** ● After the merge request is merged, update your local repository: git checkout main, git pull origin main
4	**Implement BitBucket Operations using Git.** **Task 1: Creating a Repository** ● Sign in to your Bitbucket account. ● Click the "Create" button to create a new repository. ● Choose a repository name, visibility (public or private), and other settings. ● Click "Create repository." **Task 2: Cloning a Repository** ● Open your terminal or command prompt. ● Navigate to the directory where you want to clone the repository. ● Copy the repository URL from Bitbucket. ● Run the following command: git clone <repository_url> ● Replace <repository_url> with the URL you copied from Bitbucket. ● This will clone the repository to your local machine. **Task 3: Making Changes and Creating a Branch** ● Navigate into the cloned repository: cd <repository_name> ● Create a new text file named "example.txt" using a text editor. ● Add some content to the "example.txt" file. ● Save the file and return to the command line. ● Check the status of the repository: git status

	● Stage the changes for commit: git add example.txt ● Commit the changes with a descriptive message: git commit -m "Add content to example.txt" ● Create a new branch named "feature": git branch feature ● Switch to the "feature" branch: git checkout feature **Task 4: Pushing Changes to Bitbucket** ● Add Repository URL in a variable: git remote add origin <repository_url> ● Replace <repository_url> with the URL you copied from Bitbucket. ● Push the "feature" branch to Bitbucket: git push origin feature ● Check your Bitbucket repository to confirm that the new branch "feature" is available. **Task 5: Collaborating through Pull Requests** **1. Create a pull request on Bitbucket:** ○ Go to the repository on Bitbucket. ○ Click on "Create pull request." ○ Choose the source branch ("feature") and the target branch ("main" or "master"). ○ Review the changes and click "Create pull request." **2. Review and merge the pull request:** ○ Add a title and description for the pull request. ○ Assign reviewers if needed. ○ Once the pull request is approved, merge it into the target branch. **Task 6: Syncing Changes** ● After the pull request is merged, update your local repository: git checkout main, git pull origin main
5	**Applying CI/CD Principles to Web Development Using Jenkins, Git, and Local HTTP Server** **Step 1: Set Up the Web Application and Local HTTP Server** ● Create a simple web application or use an existing one. Ensure it can be hosted by a local HTTP server. ● Set up a local HTTP server (e.g., Apache or Nginx) to serve the web application. Ensure it's configured correctly and running. **Step 2: Set Up a Git Repository** Create a Git repository for your web application. Initialize it with the following commands: git init, git add, git commit -m "Initial commit" ● Create a remote Git repository (e.g., on GitHub or Bitbucket) to push your code to later. **Step 3: Install and Configure Jenkins** ● Download and install Jenkins following the instructions for your operating system (https://www.jenkins.io/download/). ● Open Jenkins in your web browser (usually at http://localhost:8080) and complete the initial setup. ● Install the necessary plugins for Git integration, job scheduling, and webhook support. ● Configure Jenkins to work with Git by setting up your Git credentials in the Jenkins Credential Manager.

	Step 4: Create a Jenkins Job ● Create a new Jenkins job using the "Freestyle project" type. ● Configure the job to use a webhook trigger. You can do this by selecting the "GitHub hook trigger for GITScm polling" option in the job's settings. **Step 5: Set Up the Jenkins Job (Using Execute Shell)** ● In the job configuration, go to the "Build" section. ● Add a build step of type "Execute shell." ● In the "Command" field, define the build and deployment steps using shell commands. For example: # Checkout code from Git # Build your web application (e.g., npm install, npm run build) # Copy the build artefacts to the local HTTP server directory rm -rf /var/www/html/webdirectory/* cp -r * /var/www/html/webdirectory/ **Step 6: Set Up a Webhook in Git Repository** ● In your Git repository (e.g., on GitHub), go to "Settings" and then "Webhooks." ● Create a new webhook, and configure it to send a payload to the Jenkins webhook URL (usually http://jenkins-server/github-webhook/). Make sure to set the content type to "application/json." ● OR use “GitHub hook trigger for GITScm polling?” Under Build Trigger **Step 7: Trigger the CI/CD Pipeline** ● Push changes to your Git repository. The webhook should trigger the Jenkins job automatically, executing the build and deployment steps defined in the "Execute Shell" build step. ● Monitor the Jenkins job's progress in the Jenkins web interface. **Step 8: Verify the CI/CD Pipeline** Visit the URL of your local HTTP server to verify that the web application has been updated with the latest changes.
6	**Exploring Containerization and Application Deployment with Docker** **Task 1: Install Docker** ● If you haven't already, install Docker on your computer by following the instructions provided on the Docker website (https://docs.docker.com/get-docker/). **Task 2: Create a Simple HTML Page** ● Create a directory for your web server project. ● Inside this directory, create a file named index.html with a simple "Hello, Docker!" message. This will be the content served by your Apache web server. **Task 3: Create a Dockerfile** ● Create a Dockerfile in the same directory as your web server project. The Dockerfile defines how your Apache web server application will be packaged into a Docker container. Here's an example: //teacher can use other example for demo Dockerfile

	# Use an official Apache image as the base image FROM httpd:2.4 # Copy your custom HTML page to the web server's document root COPY index.html /usr/local/apache2/htdocs/ **Task 4: Build the Docker Image** ● Build the Docker image by running the following command in the same directory as your Dockerfile: docker build -t my-apache-server . ● Replace my-apache-server with a suitable name for your image. **Task 5: Run the Docker Container** Start a Docker container from the image you built: docker run -p 8080:80 -d my-apache-server ● This command maps port 80 in the container to port 8080 on your host machine and runs the container in detached mode. **Task 6: Access Your Apache Web Server** Access your Apache web server by opening a web browser and navigating to http://localhost:8080. You should see the "Hello, Docker!" message served by your Apache web server running within the Docker container. **Task 7: Cleanup** Stop the running Docker container: docker stop <container_id> ● Replace <container_id> with the actual ID of your running container. ● Optionally, remove the container and the Docker image: docker rm <container_id>, docker rmi my-apache-server
7	**Applying CI/CD Principles to Web Development Using Jenkins, Git, usingDocker Containers** **Task 1: Set Up the Web Application and Git Repository** ● Create a simple web application or use an existing one. Ensure it can be hosted in a Docker container. ● Initialise a Git repository for your web application and push it to GitHub. **Task 2: Install and Configure Jenkins** ● Install Jenkins on your computer or server following the instructions for your operating system (https://www.jenkins.io/download/). ● Open Jenkins in your web browser (usually at http://localhost:8080) and complete the initial setup, including setting up an admin user and installing necessary plugins. ● Configure Jenkins to work with Git by setting up Git credentials in the Jenkins Credential Manager. **Task 3: Create a Jenkins Job** ● Create a new Jenkins job using the "Freestyle project" type. ● In the job configuration, specify a name for your job and choose "This project is parameterized." ● Add a "String Parameter" named GIT_REPO_URL and set its default value to your Git repository URL. ● Set Branches to build -> Branch Specifier to the working Git branch (ex */master)

	● In the job configuration, go to the "Build Triggers" section and select the "GitHub hook trigger for GITScm polling" option. This enables Jenkins to listen for GitHub webhook triggers. **Task 4: Configure Build Steps** ● In the job configuration, go to the "Build" section. ● Add build steps to execute Docker commands for building and deploying the containerized web application. Use the following commands: # Remove the existing container if it exists: docker rm --force container1 # Build a new Docker image: docker build -t nginx-image1 . # Run the Docker container: docker run -d -p 8081:80 --name=container1 nginx-image1 ● These commands remove the existing container (if any), build a Docker image named "nginx-image1," and run a Docker container named "container1" on port 8081. **Task 5: Set Up a GitHub Webhook** ● In your GitHub repository, navigate to "Settings" and then "Webhooks." ● Create a new webhook, and configure it to send a payload to the Jenkins webhook URL (usually http://jenkins-server/github-webhook/). Set the content type to "application/json." **Task 6: Trigger the CI/CD Pipeline** ● Push changes to your GitHub repository. The webhook will trigger the Jenkins job automatically, executing the build and deployment steps defined in the job configuration. ● Monitor the Jenkins job's progress in the Jenkins web interface. **Task 7: Verify the Deployment** Access your web application by opening a web browser and navigating to http://localhost:8081 (or the appropriate URL if hosted elsewhere).
8	**Practical Maven Assignment** **Task 1: Setting up Maven Environment** • installing Maven on local machines and configuring it properly. • understand the directory structure and how to create a basic Maven project. **Task 2: Creating a Simple Maven Project** • create a simple Java project using Maven. • This could involve creating classes, adding dependencies, and configuring the project's POM (Project Object Model). **Task 3: Dependency Management** • Introduce dependency management in Maven. • Assign tasks such as adding external dependencies to the project using Maven Central repository and managing version conflicts. **Task 4: Build Automation** • set up automated builds using Maven. • configure Maven to compile the code, run tests, and generate artifacts like JAR files.

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-614 MJ :Soft Computing

No. of Credits:2	Teaching Scheme Theory: 2 Hrs/Week	Examination Scheme Continuous Evaluation: 15 Marks End Semester : 35 Marks

Prerequisite:

- A strong mathematical background
- Proficiency with algorithms
- Problem solving skills and critical thinking

Objectives:

- To introduce the ideas of soft computational techniques based on human experience.
- To develop the skills to design, analyze and perform experiments on real life problems using different Neural Learning Algorithms
- To conceptualize fuzzy logic and its implementation for real world applications
- To provide mathematical background to carry out optimization using genetic algorithms

Course Outcomes:

On completion of this course, student will be able to –

CO1: Learn about soft computing techniques and their applications

CO2: Analyze various neural network architectures and perceptrons

CO3: Define the fuzzy systems

CO4: Analyze the genetic algorithms and their applications.

Unit No.	Name of Unit	Teaching Hours	CO Targeted
1	**Introduction to Soft Computing**	**2**	**CO1**
	1.1 Neural Network: Definition, Advantages, Applications and scope 1.2 Fuzzy Logic: Definition, Applications 1.3 Genetic Algorithms: Definition, Applications		
2	**Neural Network**	**15**	**CO2**
	2.1:Introduction to ANN: Introduction to ANN, History of Neural Network, Structure and working of Biological Neural Network, Neural net architecture, Models of neuron-Mc Culloch & Pitts model, Perceptron, Applications of neural networks, Comparison of BNN and ANN 2.2 Learning Algorithms: Learning and Memory, Learning Algorithms, Numbers of hidden nodes, Error Correction and Gradient Decent Rules, Perceptron Learning Algorithms, Supervised		

	Learning Backpropagation, Multilayered Network Architectures, Back propagation Learning Algorithm, Feed forward and feedback neural networks, example and applications. 2.3 Associative learning: Introduction, Associative Learning, Hopfield network, Error Performance in Hopfield networks, simulated annealing, Boltzmann machine and Boltzmann learning, State transition diagram and false minima problem, stochastic update, simulated annealing. 2.4 Competitive learning Neural network : Components of CL network, Pattern clustering and feature mapping network, ART networks, Features of ART models, character recognition using ART network. Self-Organization Maps (SOM): Two Basic Feature Mapping Models, Self-Organization Map, SOM Algorithm, Properties of Feature Map, Computer Simulations, Learning Vector Quantization, Adaptive Pattern Classification 2.5 Convolution Neural Network: Building blocks of CNNs, Architectures, convolution / pooling layers, Padding, Strided convolutions, Convolutions over volumes, SoftMax regression, Deep Learning frameworks, Training and testing on different distributions, Bias and Variance with mismatched data distributions, Transfer learning, multi-task learning, end-to-end deep learning, 2.6 Application of ANN: Pattern classification – Recognition of Olympic games symbols, Recognition of printed Characters. Neocognitron – Recognition of handwritten characters. NET Talk: to convert English text to speech. Recognition of consonant vowel (CV) segments, texture classification and segmentation		
3	**Fuzzy Set Theory**	**9**	**CO3**
	3.1 Overview of Conventional Set Theory 3.2 Introduction to Fuzzy Sets, Properties of Fuzzy Sets 3.3 Operations on Fuzzy Sets, Crisp Relation, Fuzzy Relation, 3.4 Tolerance and equivalence relation, Fuzzy Tolerance and equivalence relation, 3.5 Fuzzy Max-Min and Max-Product Composition, 3.6 Membership Functions, 3.7 Fuzzification, Defuzzification to crisp sets, λ-Cuts for fuzzy Relations, 3.8 Fuzzy (Ruled-Based) system, 3.9 Graphical technique of inference, 3.10 Membership value assignment-Intuition, Inference.		

4	**Genetic Algorithms**	4	**CO4**
	4.1 Introduction to Genetic Algorithms History of Genetic Algorithms, What is Genetic Algorithms? Strengths and weaknesses of Genetic Algorithms 4.2 Traditional Optimization and Search Techniques 4.3 Basic terminologies in Genetic Algorithm 4.4 Operators in Genetic Algorithm 4.5 Simple Genetic Algorithm 4.6 Applications of Genetic Algorithms Optimization Problems, Combinational Optimization, Machine Learning, Image Processing		

Reference Books:

1. Neural Networks By Satish Kumar, Tata McGraw Hill
2. Introduction to Soft Computing by Deepa & Shivanandan, Wiley Publication
3. Fuzzy Logic With Engineering Applications by Timothy Ross, Wiley Publication
4. Genetic Algorithm in Search, Optimization and Machine Learning by David E. Goldberg, Pearson Education
5. Artificial Neural Networks - B. Vegnanarayana Prentice Hall of India P Ltd ,2005
6. Neural Networks in Computer Intelligence- Li Min Fu, MC GRAW HILL EDUCATION, 2003
7. Neural Networks -James A Freeman David M S Kapura, Pearson Education, 2004.
8. Introduction to Artificial Neural Systems- Jacek M. Zurada, JAICO Publishing House Ed.,2006.

Text Books:

1. Neural Networks a Comprehensive Foundations, Simon Haykin, PHI edition.
2. Laurene Fausett:Fundamentals of Neural Networks: Architectures, Algorithms & Apps, Pearson, 2004.
3. An introduction to neural networks, Gurney, Kevin, CRC press.

e-Books:

1.https://www.pdfdrive.com/neural-networks-a-comprehensive-foundationpdf-e18774300.html
2.https://www.pdfdrive.com/elements-of-artificial-neural-networks-e17103719.html
3.https://www.pdfdrive.com/neural-networks-methodology-and-applications-e38107895.html

MOOC Courses:

1.https://nptel.ac.in/courses/117105084
2. https://www.coursera.org/projects/predicting-weather-artificial-neural-networks

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-615-MJP: Practical on CS-614-MJ (Soft Computing)

Implement the program in C/C++/Java/ Matlab/ Python

Sr. No.	Assignments
1	Write a program to implement Fuzzy Operations Union, Intersection, Complement, Algebraic sum, Algebraic product, Cartesian product
2	Write a program to implement De Morgan's law
3	Write a program to implement Max-Min Composition and Max-Product Composition.
4	Write a program to implement lambda cut
5	Build simple Neural network in Python from scratch.
6	Build simple Neural network in Python from Keras.
7	Implementing Artificial Neural Network training process
8	Implement Back propagation
9	Implement deep learning
10	Implement Multilayer perceptron algorithm
11	Write program to create target string, starting from random string using Genetic Algorithm
12	Write program to Implement travelling salesman problem using genetic algorithm
13	Write program to study and analyze genetic life cycle

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-III

CS-631-RP : Research Work-I

No. of Credits: 4	Total Duration 120 Hours	Examination Scheme Continuous Evaluation: 30 Marks End Semester : 70 Marks

Objectives

- To acquire skills necessary for conducting independent research in the field of computer science.
- To apply theoretical concepts learned throughout the program to solve real-world problems.
- To foster collaboration and teamwork by working in groups under the guidance of a mentor.
- To enhance communication skills through presentation and documentation of research findings.

Course Outcomes

On Completion of this course, student will be able to -

CO1: Independently conduct research in a specific area of computer science

CO2: Apply appropriate research methodologies to address research problems.

CO3: Analyze and synthesize information gathered from literature reviews, experiments, or data analysis

CO4: Develop innovative solutions to research problems within the scope of computer science.

CO5: Effectively present research findings through written reports, oral presentations, or poster presentations.

CO6: Publish research work in reputable journals, present at conferences or in recognized project competitions.

Sr. No.	Guidelines for Research Project Work
1	Each student or group of students must submit a detailed project proposal outlining the research problem, objectives, methodology, and expected outcomes.
2	A mentor will be assigned by college to each group of students to provide guidance and support throughout the research process as well as to do internal assessment.
3	Students are required to conduct a thorough literature review to understand the current state of research in their chosen area.
4	Students should execute the research plan outlined in their proposal, adhering to ethical guidelines and academic standards.
5	Proper documentation of the research process, including experimental setup, data collection methods, and analysis techniques, should be maintained
6	Upon completion of the research work, students must prepare a project report and encouraged to publish their research work in reputed journals, present at conferences or in recognized project competitions to disseminate their findings.
8	Evaluation will be as per the University guidelines, based on the quality of the research work, adherence to the research plan, presentation skills, and contribution

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-IV

CS-651-MJP : Full Time Industrial Training (IT)

No. of Credits: 16	Total Duration 480 Hours	Examination Scheme Continuous Evaluation: 120 Marks End Semester : 280 Marks

Objectives

- To provide students with an opportunity to apply theoretical knowledge gained throughout the program in a real-world industrial setting
- To foster professional skills such as teamwork, communication, time management, and problem-solving in an industrial environment.
- To expose students to the practices, technologies, and challenges prevalent in the IT industry or related sectors.
- To enable students to gain hands-on experience by working on projects or tasks relevant to their field of study.
- To facilitate networking opportunities with professionals in the industry, potentially leading to future career prospects..

Course Outcomes

On Completion of this course, student will be able to –

CO1: Apply theoretical concepts learned in the classroom to solve practical problems encountered in an industrial setting.

CO2: Demonstrate proficiency in using industry-standard tools, technologies, and methodologies relevant to their area of specialization.

CO3: Apply analytical and problem-solving skills to address challenges encountered during the industrial training

CO4: Collaborate effectively with team members to achieve project goals and objectives.

CO5: Manage time and resources efficiently to complete assigned tasks and projects within the stipulated timeframe.

CO6: Prepare a comprehensive report documenting their experience, including project details, learnings, and reflections.

Sr. No.	Guidelines for Full Time Industrial Training (IT)
1	Students are required to secure an industrial/internship placement in any organization, institution, or IT industry relevant to their field of study.
2	Students must submit the offer letter from the organization within two weeks of starting the industrial training/internship, detailing the terms and duration of the internship.
3	Students must have to work full time in the organization as per their rules and regulations.
4	A mentor will be assigned to each group of students to provide guidance and support throughout the internship period.
5	The industrial training/ internship duration should span a minimum of 360 hours, equivalent to 12 credits.

6	Students may be assigned specific projects or tasks or assignments by the host organization, relevant to their area of specialization.
7	Students should provide regular updates to their mentor through progress reports time to time regarding their progress, challenges faced, and lessons learned during the industrial training.
8	Upon completion of the industrial training/ internship, students must submit a comprehensive report documenting their internship experience, including project/ assignment details, challenges and achievements as per the format specified.
9	Evaluation will be based on the quality content of the internship report, feedback from the host organization, and the overall performance during the internship/ industrial training period.

Evaluation Pattern

- Internal assessment will be carried by college guide/ mentor by continuous evaluation method.
- The final examination or presentation of the work carried during the training/internship period will be in front the panel of examiners as per the schedule given by University.
- There will be a panel of three examiners for the final assessment
  1. Industry expert (Appointed by the college)
  2. Academic expert (Appointed by the University)
  3. College guide/Mentor of the student as an internal examiner

Parameters for Evaluation	**Marks**
Internal Assessment by Mentor Regular updates, timely report submission and deliverable (40 Marks) Professional Conduct, Learning and Skill Development (40 Marks) Work Undertaken and Learning Outcomes (40 Marks)	**120**
External Assessment by Industry expert and Academic expert	**280**
Relevance and significance of the project or tasks undertaken (50 Marks) Technical proficiency demonstrated during the internship (50 Marks) Communication skills and presentation of the internship experience (50 Marks) Work Undertaken and Quality of the internship report (50 Marks) Overall performance and contribution to the organization (80 Marks)	
Total Assessment	**400**

Savitribai Phule Pune University

M.Sc.(Computer Science)

Progress Report for CS-651-MJP : Full Time Industrial Training (IT)

(This Progress report is to be submitted monthly to the college guide/Mentor)

Name of College	
Roll No./ID and Name of Student	
Date of Report Submission	
Duration of Report (From date – To date)	
Name of Organization	
Date of Joining in the organization	
Name of Industry Guide/Supervisor	
Name of College Guide/Mentor	

1. Introduction *(Mention brief overview of the internship objectives and the role of the student within the organization)*

2. Work Undertaken *(Summary of the tasks or assignments or projects undertaken by the student during the reporting period with responsibilities assigned and progress made on each task)*

3. Learning and Skill Development *(Mention the summary of new skills, knowledge, and experiences gained during the reporting period)*

4. Challenges Faced *(Put any challenges or obstacles encountered during the reporting period and Strategies adopted to overcome these challenges and lessons learned from them)*

5. Achievements and Contributions *(Highlight of notable achievements, contributions, or successes attained during the reporting period)*

6. Future Plan *(Specify future goals and objectives for the remaining period of the internship. Also put the plan of action to address any identified areas for improvement or skill enhancement)*

Signature:	Signature:
Date:	Date:
Name:	Name:
Industry Guide/Mentor	**College Guide/Mentor**

Savitribai Phule Pune University
M.Sc. Computer Science (From 2024-25) Sem-IV

CS-681-RP : Research Work-II

No. of Credits: 6	Total Duration 180 Hours	Examination Scheme Continuous Evaluation: 45 Marks End Semester : 105 Marks

Objectives

- To acquire skills necessary for conducting independent research in the field of computer science.
- To apply theoretical concepts learned throughout the program to solve real-world problems.
- To foster collaboration and teamwork by working in groups under the guidance of a mentor.
- To enhance communication skills through presentation and documentation of research findings.
- To get hands-on-experience for applying research methodology

Course Outcomes

On Completion of this course, student will be able to -

CO1: Independently conduct research in a specific area of computer science

CO2: Apply appropriate research methodologies to address research problems.

CO3: Analyze and synthesize information gathered from literature reviews, experiments, or data analysis

CO4: Develop innovative solutions to research problems within the scope of computer science.

CO5: Effectively present research findings through written reports, oral presentations, or poster presentations.

CO6: Publish research work in reputable journals, present at conferences

Sr. No.	Guidelines for Research Work
1	Each student carry out the research work during semester-IV under the guidance of mentor or guide appointed.
2	Student shall work on a research problem and publish paper(s) or article(s) or file a copyright or patent based on the work carried out. Student can also present a paper in national/international conferences.
3	Students are required to conduct a thorough literature review to understand the current state of research in their chosen area.
4	Students should execute the research plan outlined in their proposal, adhering to ethical guidelines and academic standards.
5	Proper documentation of the research process, including experimental setup, data collection methods, and analysis techniques, should be maintained
6	Upon completion of the research work, students must prepare a report
7	Evaluation will be as per the University guidelines, based on the quality of the research work, adherence to the research plan, presentation skills, and contribution