COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS601N	DCC	Compiler Design	60	20	20	30	20	2	1	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

COURSE OBJECTIVES:

The student will have ability:

1. To introduce the major concept areas of language translation and compiler design
2. To enrich the knowledge in various phases of compiler and its use
3. To provide understanding of steps of programming necessary for constructing a compiler

COURSE OUTCOMES:

Upon completion of the subject, students will be able to:

1. Ability to apply the knowledge of lex tool & yacc tool to develop a scanner & parser
2. Ability to design and develop software systems for backend of the compiler
3. Ability to comprehend and adapt to new tools and technologies in compiler design

SYLLABUS

UNIT I — 8 HOURS

Introduction: Compiler, Compilers analysis of the source program, Phases of a compiler, Cousins of the Compiler, Grouping of Phases and Compiler construction tools, Lexical Analysis, Role of Lexical Analyzer, Input Buffering and Specification of Tokens.

UNIT II — 10 HOURS

Syntax Analysis: Role of the parser, Writing Grammars, Context-Free Grammars, Top-Down parsing, Recursive Descent Parsing, Predictive Parsing, Bottom-up parsing, Shift Reduce Parsing, Operator Precedent Parsing, LR Parsers, SLR Parser – Canonical LR Parser – LALR Parser.

UNIT III — 9 HOURS

Intermediate Code Generation: Syntax Directed Definitions, Evaluation Orders for Syntax Directed Definitions, Intermediate languages, Declarations, Assignment Statements, Boolean Expressions, Case Statements, Three Address code, Back patching, Procedure calls.

UNIT IV — 9 HOURS

Code Optimization and Run Time Environments: Introduction, Principal Sources of Optimization, Optimization of basic Blocks, DAG representation of Basic Blocks - Introduction to Global Data Flow Analysis, Runtime

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS601N	DCC	Compiler Design	60	20	20	30	20	2	1	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Environments, Source Language issues, Storage Organization, Storage Allocation strategies, Access to non-local names, Parameter Passing, Error detection and recovery.

UNIT V **9 HOURS**

Code Generation: Issues in the design of code generator, The target machine, Runtime Storage management, Basic Blocks and Flow Graphs, Next-use Information, A simple Code generator, Peephole Optimization.

TEXTBOOKS:

1. A. V. Aho and J. D. Ullman, *Compilers: Principles, Techniques and Tools*. Singapore: Pearson Education Asia, 2012.
2. J. P. Tremblay and P. G. Sorenson, *The Theory and Practice of Compiler Writing*. Hyderabad, India: BS Publications, 2005.

REFERENCE:

1. D. M. Dhamdhere, *Compiler Construction: Principles and Practice*, 2nd ed. New Delhi, India: Macmillan India Ltd., 2008.
2. A. I. Holub, *Compiler Design in C*. New Delhi, India: Prentice Hall of India, 2003.
3. C. N. Fischer and R. J. LeBlanc, *Crafting a Compiler with C*. Redwood City, CA, USA: Benjamin Cummings, 2003.
4. H. Alblas and A. Nymeyer, *Practice and Principles of Compiler Building with C*. New Delhi, India: Prentice Hall of India, 2001.
5. K. C. Louden, *Compiler Construction: Principles and Practice*. Pacific Grove, CA, USA: Thomson Learning, 2003.

LIST OF PRACTICALS

1. To study the Lex Tool.
2. To study the Yacc Tool.
3. Write a program to implement Lexical Analyzer to recognize few patterns of C.
4. Write a program to implement the Recursive Descent Parser.
5. Write a program to implement the Computation of FIRST and FOLLOW of variables of grammar.
6. Write a program to compute the leading and trailing symbols of grammar.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS601N	DCC	Compiler Design	60	20	20	30	20	2	1	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

7. Write a program to implement Operator Precedence Parser.
8. Write a program to implement SLR parser.
9. Write a program to check the data types.
10. Write a program to implement the generation of three address codes.
11. Write a program to implement the computation of postfix notation.
12. Write a program to implement the computation of Quadruple.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS603N	**DCC**	**Introduction to Cloud Computing**	**60**	**20**	**20**	**30**	**20**	**3**	**0**	**2**	**4**

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

COURSE OBJECTIVES:

The student will have ability to:

1. Analyze the SAAS, PAAS IAAS services of Cloud Computing to represent how engineering agility in an organization can be created.
2. Assess the exploitation of web services from cloud computing.
3. Configure essential infrastructural components used for implementing Cloud.
4. Significantly study case studies to derive the most excellent practice model to be appropriate when deploying cloud-based applications.

COURSE OUTCOMES:

Upon completion of the subject, students will be able to:

1. Investigate the trade-offs among deploying applications in the cloud and over the local infrastructure.
2. Compute real-world problems security, privacy issues using cloud computing through group collaboration.
3. Development and Deployment applications over commercial cloud computing infrastructures.
4. Analyze and investigation of application & hardware performance, scalability, and availability of the underlying cloud technologies and software.

SYLLABUS

UNIT I **10 HOURS**

Overview of Cloud Computing: Introduction- Evolution, Shift from distributed computing to cloud computing; principles and characteristics of cloud computing- IaaS, PaaS, SaaS; service-oriented computing and cloud environment, Advantages, Service & Deployment Models, Infrastructure, and Consumer View, Functioning of Cloud Computing, Cloud Architecture, Cloud Storage, Cloud Services, Industrial Applications.

UNIT II **8 HOURS**

Cloud Computing Technology- Client systems, Networks, server systems and security from services perspectives, security and privacy issues; accessing the cloud with platforms and applications; Cloud storage

UNIT III **9 HOURS**

Working with Cloud: Infrastructure as a Service – conceptual model and working, Platform as a Service –

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS603N	DCC	Introduction to Cloud Computing	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

conceptual model and functionalities. Software as a Service –conceptual model and working. Trends in Service provisioning with clouds. Working on Microsoft Azure & IBM Smart Cloud.

UNIT IV **9 HOURS**

Using Cloud Services- Cloud collaborative applications and services – case studies with calendars, schedulers, and event management; cloud applications in project management. Amazon Web Services & applications, AWS EC2, S3, Cloud Analytics, Cloud Open Stack.

UNIT V **9 HOURS**

Case studies- Microsoft Azure, Google App Engine, IBM Smart Cloud and Open source clouds,-Open-Nebula, Sales force and Eucalyptus, Cloud Simulation.

TEXTBOOKS:

1. A. T. Velte, T. J. Velte, and R. Elsenpeter, *Cloud Computing: A Practical Approach*. New York, NY, USA: McGraw-Hill, 2010.

REFERENCE:

1. K. Saurabh, *Cloud Computing*. New Delhi, India: Wiley Publications, 2012.
2. R. L. Krutz and D. Vines, *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*. Hoboken, NJ, USA: Wiley Publications, 2013.
3. B. Sosinsky, *Cloud Computing Bible*. Hoboken, NJ, USA: Wiley Publications, 2011.
4. M. Woodside, J. Chinneck, and M. Litiou, “Adaptive cloud deployment using persistence strategies and application awareness,” *IEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 277–290, Jul.–Sep. 2017.
5. R. Buyya and C. S. Selvi, *Mastering Cloud Computing*. New Delhi, India: Tata McGraw-Hill, 2013.
6. M. Miller, *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online*. Indianapolis, IN, USA: Pearson Publishing, 2011.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS603N	DCC	Introduction to Cloud Computing	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

LIST OF PRACTICALS

1. Service deployment & Usage over cloud using VirtualBox.
2. Performance evaluation of services over cloud using VMware tool.
3. Working of Goggle Drive to make spreadsheet.
4. Working on Heroku for Cloud application deployment.
5. Working on Aneka services for Cloud application.
6. Working on services of Google App Engine.
7. Working on Application deployment & services of Microsoft Azure.
8. Working on Application deployment & services of IBM SmartCloud.
9. Working and configuration of Euceliptus.
10. Deployment & Services of Amazon Web Services.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	**SEC**	**Minor Project**	**0**	**0**	**0**	**60**	**40**	**0**	**0**	**4**	**2**

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

COURSE OBJECTIVES:

The student will have ability to:

1. Enable students to identify real-world, complex engineering problems and apply systematic research methodologies to formulate viable technical solutions.
2. Train students in designing, engineering, testing, and validating a complete product, system module, or computing architecture using modern engineering tools and industry-standard practices.
3. Foster collaborative engineering management skills, technical report writing, and the professional defense of engineered solutions before an evaluation panel.

COURSE OUTCOMES:

Upon completion of the subject, students will be able to:

1. Formulate a well-defined problem statement by critically reviewing contemporary research literature, identifying technical gaps, and establishing clear engineering requirements.
2. Design and develop an architectural framework, hardware prototype, or software system by incorporating sustainable design methodologies, algorithm design, and resource optimization.
3. Implement and deploy the proposed system using modern development environments, programming paradigms, cloud infrastructure, or advanced technical tools.
4. Evaluate the performance, scalability, and security parameters of the developed system against standard benchmark metrics, interpreting the results through rigorous testing.
5. Compile comprehensive, industry-grade technical project reports and documentation compliant with standard engineering formats, defending the work effectively through oral presentations.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	SEC	Minor Project	0	0	0	60	40	0	0	4	2

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Project Timeline

Phases

Phase 1: Planning & Requirement Analysis (Weeks 1–2)
Focus: Defining the scope, setting up infrastructure, and identifying functional requirements.

- **Week 1: Project Initiation**
  - Form teams and define the problem statement.
  - **Tooling:** Set up **Jira** boards (Scrum or Kanban), create Backlog, and initialize **Git** repositories.
  - **Deliverable:** Project Vision document and initial Product Backlog.

- **Week 2: Functional Modeling**
  - Identify Actors and Use Cases.
  - **UML Deliverables: * Use Case Diagram:** To define system boundaries.
    - **Use Case Descriptions:** Detailed text for each primary flow.
  - **Tooling:** Map Use Cases to Jira User Stories.

Phase 2: Object-Oriented Analysis (Weeks 3–4)
Focus: Understanding the "What" of the system without focusing on implementation.

- **Week 3: Static Analysis**
  - Identify domain entities and their relationships.
  - **UML Deliverables:**
    - **Object Diagram:** To model specific instances and data snapshots.
    - **Initial Class Diagram (Domain Model):** Focus on attributes and relationships (1:1, 1:N, M:N) without methods.

- **Week 4: Dynamic Analysis**
  - Model how the system reacts to events.
  - **UML Deliverable: Activity Diagram:** To model complex business logic and parallel workflows.

Phase 3: Object-Oriented Design (Weeks 5–7)
Focus: Defining the "How" – architecture, patterns, and component interactions.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	SEC	Minor Project	0	0	0	60	40	0	0	4	2

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

- **Week 5: Interaction Modeling**
  - **UML Deliverable: * Sequence Diagram:** Detailed logic showing how objects communicate over time to fulfill a use case.

- **Week 6: Detailed Class Design & Packaging**
  - Apply SOLID principles and Design Patterns.
  - **UML Deliverables:**
    - **Detailed Class Diagram:** Including methods, visibility, and interfaces.
    - **Package Diagram:** Organizing classes into logical modules/subsystems.

- **Week 7: System Architecture**
  - **UML Deliverables:**
    - **Component Diagram:** Visualizing physical modules (API, Database, UI).
    - **Deployment Diagram:** Mapping software components to hardware/nodes (Cloud, Servers).

Phase 4: Implementation & CI/CD Integration (Weeks 8–10)
Focus: Coding, version control, and automated pipelines.

- **Week 8: Sprint 1 - Core Features**
  - Develop core logic based on the Class and Sequence diagrams.
  - **Git Strategy:** Use branching (Feature branches → Develop → Main).

- **Week 9: CI/CD Pipeline Setup**
  - Configure **GitHub Actions** or **GitLab CI**.
  - Automate build triggers on every git push.

- **Week 10: Sprint 2 - Integration**
  - Develop UI and connect to the database.
  - **Jira:** Move stories to "In Progress" and "Testing."

Phase 5: Testing & Deployment (Weeks 11–12)
Focus: Quality assurance and final delivery.

- **Week 11: Testing Phase**

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	**SEC**	**Minor Project**	**0**	**0**	**0**	**60**	**40**	**0**	**0**	**4**	**2**

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

- **Unit Testing:** (JUnit, PyTest) ensuring methods work as intended.
- **Integration Testing:** Testing the flow between components.
- **Tooling:** Integrate test reports into the CI/CD pipeline.

- **Week 12: Deployment & Final Review**
  - Finalize the **Deployment Diagram** to match the production environment.
  - Deploy to a staging/production environment (Heroku, AWS, or Azure).
  - Project Demo and handover.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	SEC	Minor Project	0	0	0	60	40	0	0	4	2

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Jira Backlog and a Git Branching Strategy

1. Jira Backlog & Sprint Structure

The project is divided into **six 2-week Sprints**. **Sample Jira Epic & Story Structure:**

Epic	Sample User Stories (Tasks)	Status
Requirements & Design	Create Use Case Diagrams; Draft System Architecture; Set up Jira Board.	Done/Sprint 1
Infrastructure	Initialize Git Repo; Configure CI/CD Pipeline; Setup Database.	Sprint 2
User Management	Design User Class; Implement Login/Auth; Sequence Diagram for Auth.	Sprint 3
Core Features	[Project Specific Logic]; Unit Tests for Core Logic.	Sprint 4-5
Deployment	Final Deployment Diagram; Production Hosting; Documentation.	Sprint 6

2. Git Branching Strategy

Students should avoid pushing directly to the main branch. A **Feature Branch Workflow** is best for academic teams to prevent code conflicts.

- **main branch:** Contains stable, production-ready code.
- **develop branch:** The integration branch where all features are merged before going to main.
- **feature/feature-name branches:** Created for every individual Jira task (e.g., feature/user-login).

3. Testing & CI/CD Pipeline

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	**SEC**	**Minor Project**	**0**	**0**	**0**	**60**	**40**	**0**	**0**	**4**	**2**

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

To meet your requirement for testing software and CI/CD, the students should implement a pipeline that triggers on every **Pull Request** to the develop branch.

The Pipeline Flow:

1. **Build:** Compile the code to ensure there are no syntax errors.
2. **Unit Test:** Automatically run tests (e.g., JUnit, PyTest, or Jest). If tests fail, the merge is blocked.
3. **Static Analysis:** Use tools like SonarQube or ESLint to check for code "smells."
4. **Deploy:** Automatically deploy to a staging environment (like Vercel, Netlify, or AWS).

4. UML to Code Mapping (Object-Oriented Focus)

Students must ensure that their diagrams are not just "drawings" but blueprints:

- **Class Diagram** → Directly maps to the Classes and Attributes in their code.
- **Sequence Diagram** → Maps to the logic inside their methods (which object calls which method).
- **Package Diagram** → Maps to the Folder Structure of their project.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	SEC	Minor Project	0	0	0	60	40	0	0	4	2

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Project Grading Rubric (Total 100%)

1. Analysis & Design (30%)

- **UML Consistency:** Do the diagrams match the actual code implementation?
  - *Check:* Does the **Class Diagram** reflect the actual inheritance and associations in the source code?
- **Behavioral Accuracy:** Do **Sequence Diagrams** correctly identify method calls between objects?
- **Architectural Soundness:** Does the **Deployment Diagram** accurately map to the cloud/server infrastructure used?

2. Object-Oriented Implementation (25%)

- **Encapsulation:** Are attributes private? Are getters/setters used appropriately?
- **Inheritance/Polymorphism:** Are these used to reduce code redundancy?
- **Solid Principles:** Is the code modular? (e.g., Single Responsibility Principle).
- **Design Patterns:** Did the team implement at least one recognizable pattern (e.g., Singleton, Factory, or Observer)?

3. DevOps & Tooling (20%)

- **Jira Management:** Is the backlog updated? Are stories assigned to specific members?
  - *Check:* Do commit messages reference Jira ticket IDs?
- **Git Workflow:** Is there evidence of Pull Requests and Code Reviews?
  - *Check:* Avoid "Mega-commits" (one giant commit at the end of the week).
- **CI/CD Pipeline:** Does the pipeline automatically trigger on pushes? Is the build status visible (Green/Red)?

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	SEC	Minor Project	0	0	0	60	40	0	0	4	2

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

4. Testing & Quality (15%)

- **Unit Testing:** Is there a significant percentage of code coverage?
- **Integration Testing:** Are API endpoints or database connections tested?
- **Documentation:** Is the README.md clear, including setup instructions and a system overview?

5. Final Deployment (10%)

- **Accessibility:** Is the project hosted on a live URL?
- **Stability:** Does the system handle basic "unhappy path" errors without crashing?

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	SEC	Minor Project	0	0	0	60	40	0	0	4	2

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Definition of Done (DoD) Checklist

A Jira ticket is only considered **Done** when it meets the following criteria:

1. Design & Modeling

- **UML Alignment:** Any changes to the code structure are reflected in the **Class Diagram**.
- **Logic Validation:** For complex logic, a **Sequence Diagram** has been updated or created.
- **Documentation:** Public methods and classes are documented using standard docstrings (e.g., Javadoc, Docstring).

2. Code Quality & Standards

- **OO Principles:** Code follows Encapsulation and avoids "God Classes" (Single Responsibility Principle).
- **No Hardcoding:** Configuration values (like API keys or DB URLs) are in environment variables.
- **Peer Review:** At least one other team member has reviewed the **Pull Request (PR)** on Git.

3. Testing & CI/CD

- **Unit Tests:** New logic has corresponding unit tests.
- **CI Pipeline:** The CI/CD build is **Green** (passing) on the feature branch.
- **Zero Regressions:** Existing functionality still works as expected.

4. Integration

- **Git Merge:** The feature branch has been merged into the develop branch.
- **Jira Update:** The ticket is moved to the "Done" column and the "Actual Time Spent" is logged.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	**SEC**	**Minor Project**	**0**	**0**	**0**	**60**	**40**	**0**	**0**	**4**	**2**

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Evaluation Schedule

Phase	Milestone	Expected Artifacts
Month 1	**Foundation**	Use Case, Activity, and Object Diagrams. Jira Backlog setup.
Month 2	**Design**	Sequence, Class, and Package Diagrams. Git Feature branches active.
Month 3	**Execution**	Component and Deployment Diagrams. Working CI/CD pipeline and Unit Tests.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	SEC	Minor Project	0	0	0	60	40	0	0	4	2

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Weekly Progress Report Template

Project Title: [Project Name]

Team ID: [ID]

Date: [Date Range]

1. Executive Summary

- **Current Sprint:** [Sprint # of 6]
- **Sprint Goal:** (e.g., Finalize Sequence Diagrams and setup Database connection)
- **Status:** 🟢 On Track | 🟡 At Risk | 🔴 Delayed

2. UML & OOAD Progress

- **Diagrams Completed this week:** (e.g., Package Diagram, Component Diagram)
- **Diagrams Revised:** (List any changes made based on implementation feedback)
- **OO Concept Applied:** (e.g., "Implemented Interface Inheritance for the Payment module")

3. Technical Execution (DevOps & Git)

- **Jira Tasks Completed:** [Link to Jira tickets]
- **Git Activity:**
  - Active Branches: [List branch names]
  - Merged Pull Requests: [List PR IDs]
- **CI/CD Status:** [e.g., Build Passing / 85% Code Coverage]

4. Testing Summary

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS607N	SEC	Minor Project	0	0	0	60	40	0	0	4	2

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

- **Total Unit Tests:** [Number]
- **Pass Rate:** [Percentage]%
- **Key Bug Identified:** [Brief description of one technical challenge overcome]

5. Goals for Next Week

- [Goal A]
- [Goal B]

Project Guide Signature   Project In-charge Signature

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS710	DCC	Deep Learning Foundations	60	20	20	30	20	3	1	2	5

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

COURSE OBJECTIVES:

The student will have ability:

1. To understand the fundamental theoretical concepts of neural networks and deep learning.
2. To provide knowledge of various optimization algorithms used in training deep models.
3. To enable the design and implementation of CNNs and RNNs for real-world tasks.
4. To explore unsupervised learning frameworks like Auto encoders and Generative Models.
5. To develop skills in using modern deep learning frameworks for solving complex engineering and data-driven problems.

COURSE OUTCOMES:

Upon completion of the subject, students will be able to:

1. Demonstrate fundamental knowledge of McCulloch-Pitts neurons, Perceptron, and the mathematical foundations of gradient descent.
2. Design and optimize Multi-Layer Perceptron using backpropagation and advanced regularization techniques.
3. Implement Convolutional Neural Networks for image recognition, denoising, and computer vision tasks.
4. Apply Recurrent Neural Networks and LSTMs for sequential data analysis and natural language processing.
5. Evaluate generative models and unsupervised learning architectures for data synthesis and dimensionality reduction.

SYLLABUS

UNIT I — 9 HOURS

Foundations of Neural Networks: - History of Deep Learning, McCulloch-Pitts Neuron, Threes holding Logic, Perceptron's and Perceptron Learning Algorithm. Multilayer Perceptron's (MLPs), Representation Power of MLPs, Sigmoid Neurons Feed Forward Neural Networks, Back propagation.

UNIT II — 9 HOURS

Optimization and Linear Algebra for DL: - Gradient Descent (GD), Momentum Based GD, Nesterov Accelerated GD, Stochastic GD, AdaGrad, RMSProp, Adam. Mathematical Foundations: Eigenvalue Decomposition, Singular Value Decomposition (SVD), Principal Component Analysis (PCA).

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS710	DCC	Deep Learning Foundations	60	20	20	30	20	3	1	2	5

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

UNIT III **9 HOURS**

Regularization and Effective Training: - Bias-Variance Tradeoff, L_2 regularization, Early stopping, Dataset augmentation, Parameter sharing, Dropout, Batch Normalization, Instance and Group Normalization. Weight Initialization methods (Xavier/He), Greedy Layer wise pre-training.

UNIT IV **9 HOURS**

Convolutional Neural Networks (CNN): - Building blocks of CNN: Convolution, Pooling, and Stride. Architectures: LeNet, AlexNet, VGGNet, GoogLeNet, ResNet (Skip Connections). Applications: Image Denoising, Semantic Segmentation, Object Detection, Transfer Learning.

UNIT V **9 HOURS**

Sequence Models and Generative Modeling: - Recurrent Neural Networks (RNN), Backpropagation through time (BPTT), Vanishing/Exploding Gradients. GRU, LSTMs. Attention Mechanism and Encoder-Decoder Models. Introduction to Generative Modeling: Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN).

TEXTBOOKS:

1. I. Good fellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
2. E. Charniak, *Introduction to Deep Learning*. Cambridge, MA, USA: MIT Press, 2019.

REFERENCE:

1. F. Chollet, *Deep Learning with Python*. Shelter Island, NY, USA: Manning Publications, 2021.
2. S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2009.
3. A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensor Flow*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.
4. https://onlinecourses.nptel.ac.in/noc26_cs66/preview
5. https://onlinecourses.nptel.ac.in/noc26_ee53/preview

LIST OF PRACTICALS

1. Implementation of a McCulloch-Pitts Neuron to simulate basic logic gates.
2. Implementation of the Perceptron Learning Algorithm for binary classification.
3. Design a Multi-Layer Perceptron (MLP) for the MNIST handwritten digit dataset using Back propagation.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTCS710	**DCC**	**Deep Learning Foundations**	**60**	**20**	**20**	**30**	**20**	**3**	**1**	**2**	**5**

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

4. Comparative study of different optimizers (SGD, RMSProp, Adam) on a deep neural network.
5. Implementing Dimensionality Reduction using Auto encoders and comparing it with PCA.
6. Applying Dropout and Batch Normalization to a deep network to analyze performance and over fitting.
7. Image Classification using a standard CNN architecture (e.g., LeNet or AlexNet).
8. Implementing Transfer Learning using a pre-trained model (VGG16 or ResNet) for a custom image dataset.
9. Time-series prediction (e.g., stock price or weather) using LSTMs.
10. Implementation of a basic Generative Adversarial Network (GAN) to generate synthetic images.
11. Capstone Project

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE620	DSE	TinyML & Edge AI	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

COURSE OBJECTIVES:

The student will have ability:

1. To understand what Edge AI is and why running smart models on tiny gadgets is better than sending data to the cloud.
2. To learn how to compress a standard AI model so it can fit onto small, cheap computer chips.
3. To build real, physical projects that can see, hear, and feel movements without needing the internet.

COURSE OUTCOMES:

Upon completion of the subject, students will be able to:

1. Explain the basic differences between a computer, a smartphone, and a tiny microcontroller chip.
2. Shrink a Python-trained AI model down to a tiny fraction of its size using basic optimization tricks.
3. Program a smart physical gadget to detect voices, gestures, or objects completely offline.

SYLLABUS

UNIT I — 8 HOURS

Introduction: Introduction to Edge AI and Tiny Machine Learning (TinyML), Cloud Computing vs. Edge Computing vs. TinyML Benefits of On-Device Processing: Privacy, Latency, Bandwidth, and Power, Introduction to Microcontrollers (MCUs) vs. CPUs and GPUs, Hardware Constraints: Millivolt Power Budgets, Flash Memory, and Static RAM (SRAM), Overview of Low-Power Hardware Ecosystems (ARM Cortex-M, ESP32, Arduino Nano)

UNIT II — 10 HOURS

TinyML Software Pipeline: The Lifecycle of a TinyML Application, Specialized Edge Software Frameworks (TensorFlow Lite for Microcontrollers, Edge Impulse), Model Compilation: Converting Desktop Models (.h5/.pt) to Flatbuffer Binaries (.tflite), Converting Binaries to Static C-Byte Arrays (.cc format) for Bare-Metal Deployment, Interpreter-Free Code Generation Platforms, Static Memory Allocation and Memory Workspace Safety

UNIT III — 9 HOURS

Model Optimization and Compression Techniques: Floating-Point Arithmetic (32-bit Float) vs. Integer Constraints (8-bit Integer), Post-Training Quantization (PTQ): Dynamic Range, Full Integer, and Float16 Quantization, Quantization-Aware Training (QAT) Basics, Mathematical Core of Quantization: Scales, Offsets, and Zero-Points, Weight Pruning: Structural vs. Unstructural Matrix Thinning, Knowledge Distillation and Weight

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore

Shri Vaishnav Institute of Information Technology

Choice Based Credit System (CBCS) in the light of NEP-2020

B. Tech (CSE/ IT) - All Programs

SEMESTER-VI (2024-28)

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE620	DSE	TinyML & Edge AI	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Clustering Basics.

UNIT IV — 9 HOURS

Sensory Inputs and Signal Processing at the Edge: Interfacing and Sampling Raw Data from Low-Power Physical Sensors, Motion Processing: Multi-Axis Accelerometer and Gyroscope Stream Ingestion, Audio Processing: Time-Domain Waveforms to 2D Spectrograms via Fast Fourier Transforms (FFT), Mel-Frequency Cepstral Coefficients (MFCC) Extraction for Edge Audio, Vision Processing: Low-Resolution Camera Frame Buffering, Color-Space Conversions (RGB to Grayscale) and Downsampling for Tiny Vision Models.

UNIT V — 9 HOURS

Real-World Applications, Edge MLOps, and Security: Industrial Use Cases: Vibration Anomaly Detection and Predictive Maintenance, Environmental and Consumer Use Cases: Smart Agriculture Sensors and Wearable Trackers, Introduction to Edge MLOps: Remote Firmware Updates and Over-The-Air (OTA) Deployments, Managing Performance Degradation and Device Drift in Changing Environments, Physical Security Vulneracies of Edge Endpoints, Data Privacy and Prompt Guardrails for Local Micro-Inference Loops.

TEXTBOOKS:

1. P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2020.
2. G. M. Iodice, *TinyML Cookbook: Combine Artificial Intelligence with Microcontrollers to Create Smarter Edge Devices*, 2nd ed. Birmingham, UK: Packt Publishing, 2024. .

REFERENCE:

1. B. Fowler, *Hands-on AI Agents and Edge Projects with Python*, 1st ed. Boston, MA, USA: O'Reilly Media, 2025.
2. J. Williams, *Generative AI and Edge Blueprint: Designing Tools and Workflows*, New York, NY, USA: McGraw-Hill, 2025.
3. A. Patel, *Introduction to Embedded Systems and TinyML Workflows*, San Francisco, CA, USA: Morgan Kaufmann, 2025.
4. Google LLC, "TensorFlow Lite for Microcontrollers Guide," 2026. [Online]. Available: https://www.tensorflow.org/lite/microcontrollers

Chairperson	Chairperson	Controller of Examination	Registrar
Board of Studies, Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore	Faculty of Studies, Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore	Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore	Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE620	DSE	TinyML & Edge AI	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

5. Edge Impulse Inc., "TinyML Course and Tutorials for Beginners," 2026. [Online]. Available: https://docs.edgeimpulse.com/
6. Harvard John A. Paulson School of Engineering, "Introduction to TinyML (CS249r) Open Resources," 2025. [Online]. Available: https://tinyml.harvard.edu/

LIST OF PRACTICALS

1. The Math Magic (Sine Wave Project): Train a basic Python model to draw a perfect curve, convert it into an 8-bit integer file, and view it as raw C-code.
2. The Model Shrinker: Take an AI model and use a simple slider tool or script to reduce its size from megabytes to kilobytes. Observe the size difference!
3. The Data Collector: Shake a hardware sensor (or your smartphone) to collect data for three actions: a wave, a circle, and a rest state. Export this as a clean dataset.
4. The Gesture Detector: Put your custom motion data into a simple builder, train it, and make an on-board LED light flash blue when you wave your hand.
5. Visualizing Sound: Record a 2-second audio file of yourself speaking and use a Python script to instantly transform it into a frequency image (Spectrogram).
6. The Voice Switch: Load a small audio model onto a device. Make an LED turn ON when you say "Go" and turn OFF when you say "Stop" entirely offline.
7. Is Someone There? (Person Detector): Hook up a tiny dollar-store camera to a microcontroller and deploy an AI model that flashes a red light if a human steps into frame.
8. The Speedometer Test: Write a simple line of code to count exactly how many milliseconds your tiny chip takes to make a single prediction.
9. Battery-Saver Mode: Program your device to "sleep" when nothing is happening, waking up only when the sensor detects a sharp movement.
10. Final Capstone Build: Assemble a quick, practical prototype (e.g., a "Smart Mailbox" that blinks when mail is dropped in, or a "Pet Feed Detector"). Present a short demo video of your working gadget.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE621	DSE	Agentic AI & Multi-Agent Systems	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

COURSE OBJECTIVES:

The student will have ability:

1. To understand the transition from traditional, prompt-based LLM interactions to autonomous, goal-oriented AI agents.
2. To learn how to equip AI models with external tools, APIs, and basic memory structures to solve real-world problems.
3. To master the basics of stateful agent tracking and visual workflow design using beginner-friendly graph concepts.
4. To explore how multiple AI agents can collaborate, take on distinct roles, and communicate within a team framework.
5. To understand practical methods for testing, debugging, securing, and deploying simple AI agent applications.

COURSE OUTCOMES:

Upon completion of the subject, students will be able to:

1. Differentiate between passive LLMs and autonomous agents and construct basic prompt-based reasoning loops.
2. Create functional single agents that interact safely with web APIs, databases, and local file systems.
3. Build controllable, step-by-step agent workflows using basic state-management and human-approval checks.
4. Design a collaborative multi-agent squad where individual agents work together to execute a multi-step task.
5. Troubleshoot agent loops, measure execution costs, and apply simple guardrails to prevent unexpected behaviors.

SYLLABUS

UNIT I — 8 HOURS

Introduction to Agents & Prompt-Based Reasoning: What is an AI Agent? The basic loop: Perception → Reasoning → Action. Limitations of standard LLMs. Introduction to the ReAct (Reason + Action) pattern, Structured reasoning patterns: Chain-of-Thought (CoT), Self-Reflection (asking the model to check its own work). Introduction to Tool Calling: How an LLM decides when and how to call a simple function.

UNIT II — 10 HOURS

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE621	DSE	Agentic AI & Multi-Agent Systems	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Empowering Agents with Tools and Basic Memory: Understanding Tools: Wrapping Python functions into tools an LLM can understand. Connecting agents to external APIs (Weather APIs, Google Search, Wikipedia). Introduction to Agent Memory: The difference between short-term (current conversation) and long-term memory. Basic Vector Stores and Simple Retrieval-Augmented Generation (RAG) for localized document access. Managing context window: Memory Overflow.

UNIT III **9 HOURS**

Introduction to Stateful Workflows and Flow Control: The danger of running agents on "autopilot": infinite loops, runaway API costs, and erratic behavior. Introduction to State Management: Keeping track of what an agent has done, step-by-step. Foundations of Graph-Based Workflows: Understanding Nodes (actions) and Edges (decisions/paths). Building basic state machines using LangGraph for structured, non-linear workflows. Human-in-the-Loop Basics: How to pause an agent and wait for explicit human approval before proceeding

UNIT IV **9 HOURS**

Foundations of Multi-Agent Collaboration: Mulitple Agents, Role-based multi-agent design. Introduction to Multi-Agent Frameworks (CrewAI and AutoGen/AG2 basics), Communication Styles: Sequential task hand-offs (Agent A passes to Agent B) vs. Group Chats (conversational debate), Common Multi-Agent challenges: Token consumption, message looping, and handling conflicting instructions between agents.

UNIT V **9 HOURS**

Testing, Safety, and Deploying Your Agent: Basic evaluation**,** tracking operational metrics: Counting input/output tokens and tracking API costs per run, Introduction to Agent Safety: Simple guardrails to block offensive inputs or unintended tool execution, Preventing prompt injection: Keeping users from hijacking your agent's system instructions, Building a clean web interface for your agent using beginner-friendly tools like Streamlit and deploying it locally using FastAPI.

TEXTBOOKS:

1. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson, 2020.
2. B. Fowler, *Hands-on AI Agents: A Practical Guide to Building Intelligent Systems with Python*, 1st ed. Boston, MA, USA: O'Reilly Media, 2025.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE621	DSE	Agentic AI & Multi-Agent Systems	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

REFERENCE:

1. Williams, *Generative AI Blueprint: Designing Tools, Memory, and Workflows with Open Source Frameworks*, New York, NY, USA: McGraw-Hill, 2025.
2. A. Patel, *Introduction to LLM Engineering: Prompts, RAG, and Agentic Workflows*, San Francisco, CA, USA: Morgan Kaufmann, 2025.
3. LangChain Inc., "LangGraph Conceptual Guides for Beginners," 2026. [Online]. Available: https://langchain-ai.github.io/langgraph/concepts/
4. DeepLearning.AI, "AI Agents in LangChain & Multi-Agent Systems Short Courses," 2025. [Online]. Available: https://www.deeplearning.ai/

LIST OF PRACTICALS

1. Prompt Engineering to Agent Loop: Write a native Python loop that takes user input, sends it to an LLM using a structured prompt template, parses the model's intended action, and prints out the "thought process."
2. Building a ReAct Loop from Scratch: Implement a simple calculator agent loop without external frameworks. The agent must parse whether the user wants to add or multiply, call a local Python math function, and return the final answer.
3. LangChain Quickstart: Build a basic search assistant using LangChain that reads a user query, automatically decides to use a provided Wikipedia tool, and delivers a clean summary.
4. Creating Custom Local Tools: Write a custom Python tool that reads, writes, or edits a local text file based on an LLM agent's commands.
5. Conversational Agent with Memory: Build a chat assistant that retains conversation history across multiple turns using LangChain's memory utilities.
6. My First RAG Agent: Load a short university policy PDF into a local vector store (e.g., ChromaDB) and construct an agent that answers student queries by pulling relevant text snippets first.
7. The Runway Interrupter: Write an agent loop that automatically stops and alerts the user if it has tried to call tools more than 5 times without finding an answer, preventing infinite loops.
8. Basic LangGraph Workflow: Create a simple 3-node graph workflow (Input Node →Processing Node →Output Node) using LangGraph to process a user's request.
9. The Human-Gatekeeper Node: Implement an approval workflow using LangGraph where an agent wants to draft an email but pauses execution until a human types "Y" in the terminal to authorize sending.
10. The Content Crew: Build a 2-agent team using CrewAI. Agent 1 (Researcher) scrapes information on a topic

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE621	DSE	Agentic AI & Multi-Agent Systems	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

and saves it; Agent 2 (Writer) reads the research and drafts a blog post.

11. The Debate Team: Build a simple two-agent conversation using AutoGen/AG2 where a "Code Generator" agent writes a script and a "Code Reviewer" agent checks it for basic syntax errors, passing feedback back and forth.
12. The Mini Software Team: Implement a 3-agent pipeline (Planner → Coder → Tester) where tasks flow sequentially from one specialized agent to the next to build a basic Python function.
13. Token Counter and Cost Estimator: Write a wrapper function around your agent that calculates and displays the exact cost of each interaction based on current LLM API pricing.
14. Implementing Simple Guardrails: Write a Python validation layer that scans an agent's output for sensitive or blacklisted words before showing it to the user.
15. The Streamlit Agent Dashboard: Build an interactive, user-friendly frontend dashboard using Streamlit that displays your agent's conversational UI along with a side panel showing its internal "Thoughts and Tool Calls."
16. Final Capstone Presentation: Package an end-to-end single or multi-agent application (e.g., a smart personal study assistant or an automated social media planner) with a working local UI, a basic safety filter, and a short presentation.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE622	DSE	Blockchain and Distributed Systems	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

COURSE OBJECTIVES:

The student will have ability:

1. To introduce the foundational concepts of distributed systems, including transparency, system models, and classical synchronization challenges.
2. To understand the structural components of blockchain networks, including cryptographic primitives, immutability, and distributed ledgers.
3. To design, write, deploy, and debug secure decentralized applications (DApps) and smart contracts using contemporary industry frameworks.

COURSE OUTCOMES:

Upon completion of the subject, students will be able to:

1. Contrast centralized, distributed, and decentralized network architectures while identifying classical failure modes and consistency issues.
2. Explain the underlying mechanics of cryptographic hashing, digital signatures, consensus protocols (PoW, PoS), and state transitions in block structures.
3. Construct, test, and programmatically interface with autonomous smart contracts to execute transactions and query immutable states.

SYLLABUS

UNIT I — 8 HOURS

Introduction: Introduction to Distributed Systems: goals, characteristics, and hardware/software architectures. The spectrum of networks: Centralized vs. Distributed vs. Decentralized topologies. Fundamental challenges: the CAP Theorem (Consistency, Availability, Partition tolerance) and FLP Impossibility. Concurrency and Synchronization: Physical clocks, Logical Clocks (Lamport Timestamps), and Vector Clocks. Understanding system failures: Crash-stop vs. Byzantine faults.

UNIT II — 10 HOURS

Classical Consensus and Cryptographic Building Blocks: Introduction to classical distributed consensus: Paxos and Raft algorithms. Fundamental Cryptography for Blockchains: Cryptographic hash functions (SHA-256 properties), collision resistance, and preimage resistance. Data integrity verification: Merkle Trees and Merkle Proofs. Identity and authentication: Asymmetric Key Cryptography (Public/Private key pairs) and the Digital

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE622	DSE	Blockchain and Distributed Systems	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Signature Algorithm (ECDSA).

UNIT III 9 HOURS

Introduction to Blockchain & Distributed Ledgers: What is a Blockchain? Detailed block anatomy: header, nonce, difficulty targets, timestamp, and transaction root. Ledger properties: immutability, append-only architecture, and transparency. Blockchain categorizations: Public, Private, and Permissioned networks (Ethereum vs. Hyperledger Fabric). Modern Decentralized Consensus mechanisms: Proof of Work (PoW) mining loops vs. Proof of Stake (PoS) validator pools.

UNIT IV 9 HOURS

Smart Contracts and Decentralized Applications: Evolution of programmable block networks: Bitcoin script vs. Ethereum Virtual Machine (EVM). Introduction to Smart Contracts: life cycle, execution models, and state variables. Programming Language Essentials: Fundamentals of Solidity (variables, visibility modifiers, control structures, mappings, functions, and events). Gas mechanics: execution costs, base fees, priority fees, and preventing out-of-gas termination loops.

UNIT V 9 HOURS

Enterprise Solutions, Security, and Future Web3 Trends: Tokenization and standard interfaces: Fungible assets (ERC-20) vs. Non-Fungible Tokens (ERC-721 / ERC-1155). Introduction to Permissioned Blockchains: Hyperledger Fabric architecture, channels, and chaincode configurations. Smart Contract vulnerabilities and security: Reentrancy bugs, arithmetic overflows, and front-running risks. Future Directions: Layer-2 scaling (Rollups, State Channels), cross-chain interoperability protocols, and decentralized storage systems (IPFS).

TEXTBOOKS:

1. I. Bashir, *Mastering Blockchain: Deeper Insights into Decentralization, Cryptography, Smart Contracts, and Distributed Ledgers*, 4th ed. Birmingham, UK: Packt Publishing, 2023.
2. A. S. Tanenbaum and M. Van Steen, *Distributed Systems: Principles and Paradigms*, 4th ed. Amsterdam, Netherlands: distributed-systems.net, 2023.

REFERENCE:

1. A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*, 1st ed.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE622	DSE	Blockchain and Distributed Systems	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

Sebastopol, CA, USA: O'Reilly Media, 2018.

2. M. Swan, *Blockchain: Blueprint for a New Economy*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2015.
3. C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to Reliable and Secure Distributed Programming*, 2nd ed. Berlin, Germany: Springer, 2011.

LIST OF PRACTICALS

1. The Hash Machine: Write a Python script to compute the SHA-256 hashes of varying text inputs. Demonstrate how modifying a single character changes the output hash entirely (Avalanche Effect).
2. Merkle Tree Constructor: Implement a functional Merkle Tree structure in Python. Take an array of transaction strings, hash them sequentially into pairs, compute the Merkle Root, and generate a validation proof for a target transaction.
3. Cryptographic Identity Workshop: Generate an asymmetric cryptographic key pair using Python or a CLI utility. Write a short text payload, sign it using the private key, and write a verification routing check using the public key.
4. My First Blockchain: Build an abstract 3-node blockchain prototype in Python. Each block must link back to the previous block's hash, keep an array of records, and recalculate its own hash validly.
5. Remix IDE Environment Setup: Initialize a workspace using the online Remix IDE. Write and execute a fundamental "Hello World" smart contract containing read/write global string parameters, observing storage variable changes.
6. State and Access Controller: Write a Solidity contract representing a simple university grading index. Restrict execution patterns using custom structural statements (e.g., standard require or modifier wrappers) so only an assigned owner address can log scores.
7. The Bank Contract & Gas Tracer: Create a contract in Solidity mimicking a simplified digital vault where users can track personal balances, deposit assets, and withdraw values safely. Analyze and record the execution gas fees of each transaction type.
8. Token Deployment (ERC-20 Prototype): Implement and deploy a custom, standards-compliant ERC-20 utility token contract onto a local virtual blockchain testing network (e.g., Hardhat, Foundry, or Ganache).
9. DApp Interfacing Lab (Web3.js / Ethers.js): Build a basic local HTML webpage interface. Connect the client-side system to a Web3 wallet extension (e.g., MetaMask) to query on-chain block numbers and pull live data metrics from your deployed token contract.
10. Capstone Decentralized Application Prototype: Build an end-to-end decentralized application tracking

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTDSE622	**DSE**	**Blockchain and Distributed Systems**	**60**	**20**	**20**	**30**	**20**	**3**	**0**	**2**	**4**

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

system (e.g., an unalterable product supply chain log or a decentralized voting ballot box). Deliverables include clear, audited smart contract components, a working client browser interface, and a localized execution walkthrough demonstration.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTMA610	DCC	UI & UX Programming	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

COURSE OBJECTIVES:

The student will have ability to:

1. Understand the fundamental concepts, principles, and importance of User Interface (UI) and User Experience (UX) design in digital products.
2. Explain user-centered design methodologies, usability principles, and accessibility guidelines used in interface design.
3. Apply design thinking techniques, wireframing, prototyping, and visual design principles to create user-friendly interfaces.
4. Analyse user requirements, user behaviour, and usability testing results to improve interface effectiveness and user satisfaction.
5. Design and develop interactive UI/UX prototypes for web or mobile applications using modern design tools and techniques.

COURSE OUTCOMES:

Upon completion of the subject, students will be able to:

1. Describe UI/UX principles, design processes, usability concepts, and accessibility standards used in digital product design.
2. Develop wireframes, prototypes, and user interfaces by applying design principles, visual hierarchy, and interaction design techniques.
3. Design and evaluate user-centered digital interfaces through usability testing and iterative design improvements.

SYLLABUS

UNIT I : 9 HOURS

Introduction to UI and UX Design: Introduction to UI and UX, Difference between UI and UX, Importance of UI/UX in digital products, User-centered design concepts, Design thinking process, Overview of UI/UX design workflow.

UNIT II 9 HOURS

User Research and Information Architecture: Understanding users and stakeholders, User research methods, Surveys, interviews, and observations, User personas and empathy maps, User journey mapping, Information architecture fundamentals, Content organization and navigation design.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTMA610	DCC	UI & UX Programming	60	20	20	30	20	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

UNIT III **9 HOURS**

Wireframing and Visual Design Principles: Introduction to wireframes, Low-fidelity and high-fidelity wireframes, Layout design principles, Typography fundamentals, Color theory and color psychology, Visual hierarchy, Grid systems and responsive layouts.

UNIT IV **9 HOURS**

Interaction Design and Prototyping: Principles of interaction design, Navigation patterns, User interactions and feedback, Designing forms and controls, Mobile-first design concepts, Responsive design principles, Interactive prototyping.

UNIT V **9 HOURS**

Usability, Accessibility, and UI/UX Project: Introduction to usability testing, Heuristic evaluation, Accessibility principles and standards, Inclusive design concepts, UX metrics and evaluation techniques, Gathering and analysing user feedback.

TEXTBOOKS:

1. B. Unger and C. Chandler, *A Project Guide to UX Design: For User Experience Designers in the Field or in the Making*, 3rd ed. Berkeley, CA, USA: New Riders, 2021.
2. D. Norman, *The Design of Everyday Things*, Revised and Expanded ed. New York, NY, USA: Basic Books, 2013.

REFERENCE:

1. S. Krug, *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*, 3rd ed. Berkeley, CA, USA: New Riders, 2014.
2. J. Garrett, *The Elements of User Experience: User-Centered Design for the Web and Beyond*, 2nd ed. Berkeley, CA, USA: New Riders, 2010.
3. A. Cooper, R. Reimann, D. Cronin, and C. Noessel, *About Face: The Essentials of Interaction Design*, 4th ed. Indianapolis, IN, USA: Wiley, 2014.

COURSE CODE	CATEGORY	COURSE NAME	TEACHING & EVALUATION SCHEME					L	T	P	CREDITS
			THEORY			PRACTICAL					
			END SEM University Exam	Two Term Exam	Teachers Assessment*	END SEM University Exam	Teachers Assessment*				
BTMA610	**DCC**	**UI & UX Programming**	**60**	**20**	**20**	**30**	**20**	3	0	2	4

Legends: **L** - Lecture; **T** - Tutorial/Teacher Guided Student Activity; **P** – Practical; **C** - Credit; ***Teacher Assessment** shall be based following components: Quiz/Assignment/ Project/Participation in Class, given that no component shall exceed more than 10 marks.

LIST OF PRACTICALS

1. Installation and familiarization with Figma or Adobe XD.
2. Create user personas and empathy maps for a selected application.
3. Develop a user journey map for an e-commerce or educational website.
4. Design a sitemap and user flow diagram for a mobile application.
5. Create low-fidelity wireframes for a web or mobile interface.
6. Design high-fidelity wireframes using a UI design tool.
7. Apply typography, color schemes, and visual hierarchy to an interface design.
8. Develop an interactive prototype with navigation and user interactions.
9. Conduct usability testing and prepare a usability evaluation report.
10. Design and present a complete UI/UX prototype for a web or mobile application.