UNIVERSITY OF BUEA

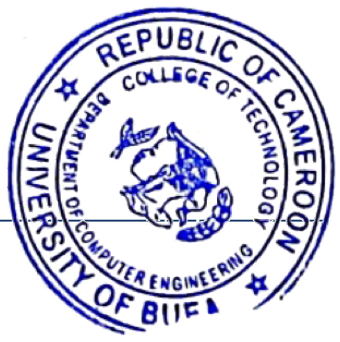
College of Technology

Department of Computer Engineering

SYLLABUS

Academic Year 2026/2027

Undergraduate and Postgraduate Programmes in
Software Engineering & Computer Networks



Head of Department
Prof. Moffo Lonla Bertrand

Table of Contents

I DEPARTMENT OVERVIEW 2

II STUDENT PROFILES & ADMISSION REQUIREMENTS 3
- II-A Option: Software Engineering . . . 3
- II-B Option: Computer Networks and Systems Maintenance . . . 3

III PROGRAM STRUCTURE — OPTION: SOFTWARE ENGINEERING 6
- III-A Year One — Level 200 (1 Credit = 15 Hours) . . . 6
- III-B Year Two — Level 300 (1 Credit = 15 Hours) . . . 6
- III-C Year Three — Level 400 (1 Credit = 15 Hours) . . . 7

IV PROGRAM STRUCTURE — OPTION: COMPUTER NETWORKS 8
- IV-A Year One — Level 200 (1 Credit = 15 Hours) . . . 8
- IV-B Year Two — Level 300 (1 Credit = 15 Hours) . . . 8
- IV-C Year Three — Level 400 (1 Credit = 15 Hours) . . . 9

V TOP-UP PROGRAMME — REMEDIAL COURSES (HND Students) 10
- V-A Software Engineering Option . . . 10
- V-B Computer Networks Option . . . 10

VI POSTGRADUATE PROGRAMME — M.TECH IN SOFTWARE ENGINEERING 11
- VI-A Year Four (1 Credit = 10 Hours) . . . 11
- VI-B Year Five (1 Credit = 10 Hours) . . . 11

VII COURSE DESCRIPTIONS — LEVEL 200 12

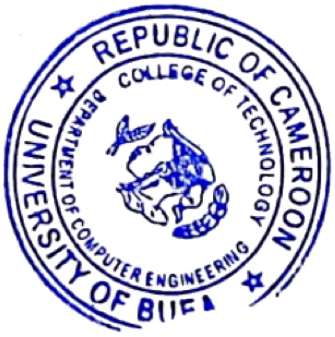
VIII COURSE DESCRIPTIONS — LEVEL 300 18

IX COURSE DESCRIPTIONS — LEVEL 400 26

X ELECTIVE COURSE DESCRIPTIONS 39

XI COURSE DESCRIPTIONS — LEVEL 600 (M.TECH) 46

XII COURSES OFFERED TO OTHER DEPARTMENTS 54



I. DEPARTMENT OVERVIEW

Mission Statement

The Department of Computer Engineering of the College of Technology, University of Buea is committed to the training of Computer Engineering Experts. The Department seeks to provide quality instruction at undergraduate level and encourages hands-on learning that enables graduates to harness forces and exploit resources as instruments for development. Undergraduate courses emphasize the teaching of basic Engineering principles and applications, enhanced by practical sessions.

Graduates are expected to:

- Think critically and strategically to solve the majority of problems in their area of expertise
- Upgrade rapidly their knowledge, skills and attitudes in a changing and challenging environment
- Assemble knowledge from different sources presented in varied perspectives
- Understand sophisticated techniques in system analysis and design
- Have a sound understanding of computer science
- Understand current programming languages such as Python, C/C++, Visual Basic and Java
- Understand the methodology of software systems engineering and software development in modern environments
- Continue their professional development throughout their career by combining theory with its application in Computer Engineering practice

Program Objectives

At the end of their training, each graduate will be able to:

- Be well-educated in the fundamental concepts of Software Engineering and Computer Networks
- Continue their professional development throughout their career by combining theory with application
- Be equipped with human resource capacity in the Software Engineering discipline in both public and private sectors
- Work as a team to improve and share all acquired knowledge
- Act as an Information Technology solutions provider

Language of Instruction

The language of instruction is as defined by the decree creating and organizing University of Buea.

Program Options

The Computer Engineering Department offers two options:

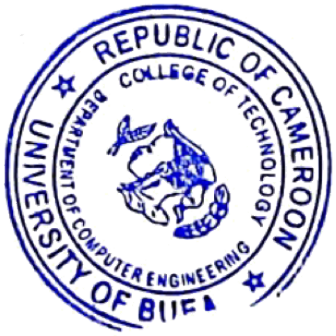
- Software Engineering (Undergraduate and Postgraduate level): Students are trained to specialize in E-Activities and Software Development.
- Computer Networks and Systems Maintenance (Undergraduate level): Students build human resource capacity in the Computer Network and system maintenance discipline to become proficient in developing Computer networks in a variety of platforms.

Head of Department

Prof. Moffo Lonla Bertrand, Associate Professor

Permanent Academic Staff

- Dr. Melingui Melono Robert
- Mr. Tatsachou Ngningaye Dieudenort
- Mr. Collins Baloko Molimi
- Mr. Anumoh Michael
- Mr. Megoze Hilaire
- Mr. Kome Kole
- Dr. Tchapga Tchito Christian



II. STUDENT PROFILES & ADMISSION REQUIREMENTS

II-A. Option: Software Engineering

Undergraduate — Admission Requirements (Year 1)

In addition to the general university requirements, candidates must possess any of the following:

- GCE A Level with at least two subjects and a minimum of Grade D in Computer Science, Mathematics, Physics, or Chemistry
- Baccalaureat C, D, E or TI
- Baccalaureat Technique (F2, F3)
- GCE A Level Technical

Undergraduate — Admission Requirements (Year 3 / Direct Entry)

- Holding an HND, DIPET I or a BTS in Software Engineering or related fields

Undergraduate — Graduation Requirements

For the award of Bachelor of Technology (B.Tech) in Software Engineering, students must complete University course requirements, core courses, complementary courses and professional courses amounting to at least 180 credits.

Postgraduate — Admission Requirements (M.Tech in Software Engineering)

- Bachelor degree in Computer Science
- Professional degree in Computer Science

Postgraduate — Graduation Requirements

For the award of Master of Technology (M.Tech) in Software Engineering, students must complete university course requirements, core courses, complementary courses and professional courses amounting to at least 120 credits.

Employment Opportunities — Software Engineering

- Technical staff in industries related to Software Engineering (software developer, programmer, web application developer, software engineer, analyst, specialist)
- Chief Executive Officer (CEO) in a small and medium-sized company
- Teacher in secondary school for Software Engineering and ICT at O Level and A Level

II-B. Option: Computer Networks and Systems Maintenance

Undergraduate — Admission Requirements (Year 1)

- GCE A Level with at least two subjects and a minimum of Grade D in Computer Science, Mathematics, Physics, or Chemistry
- Baccalaureat C, D, E or TI
- Baccalaureat Technique (F2, F3)
- GCE A Level Technical

Undergraduate — Admission Requirements (Year 3 / Direct Entry)

- Holding an HND, DIPET I or a BTS in Computer Networks or related fields

Undergraduate — Graduation Requirements

For the award of Bachelor of Technology (B.Tech) in Computer Networks and Systems Maintenance, students must complete University course requirements, core courses, complementary courses and professional courses amounting to at least 180 credits.

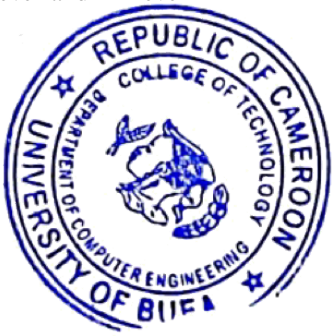
Employment Opportunities — Computer Networks

- Technical staff in industries related to Computer Networking and Systems Maintenance
- Chief Executive Officer (CEO) in a small and medium-sized company
- Teacher in secondary school for Computer Networks and System Maintenance at O Level and A Level

Compulsory Courses for Undergraduate

University Courses

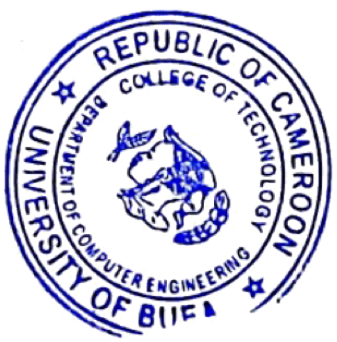
- **SPT100** — Sports
- **ENG100** — Use of English
- **FRE100** — Functional French



- **CEE100** — Civics and Ethics
- **ENT100** — Entrepreneurship
- **LAW100** — Legal Aspects of Business
- **ICT100** — Computer for Business

Departmental Courses

- **CEC209** — Introduction to Networking and Security
- **CEC213** — Fundamentals of Programming 1
- **CEC217** — Computer Architecture
- **CEC221** — Static Web Design
- **CEC223** — Algorithm and Complexity
- **COT201** — Workshop Practice 1
- **EEC209** — Mathematics 1
- **CEC218** — Introduction to Data Analysis and Machine Learning
- **CEC220** — Principles of Operating Systems and Compilers
- **CEC224** — Software Architecture
- **CEC226** — Script Language
- **COT210** — Industrial Experience 1
- **EEC204** — Mathematics 2
- **CEC305** — Data Structure and Algorithm
- **CEC315** — Introduction to Cloud Computing
- **CEC317** — Database I
- **CEC319** — Dynamic Web Design
- **CEC321** — Programming with UML
- **CEC323** — Introduction to Artificial Intelligence
- **CEC327** — Python for Networking
- **CEC329** — Computer Networks I
- **COT305** — Mathematics 3
- **COT309** — Workshop Practice 2
- **CEC302** — Introduction to Object Oriented Programming
- **CEC304** — Data Security and Integrity
- **CEC306** — System Analysis and Design
- **CEC318** — Introduction to Mobile Application Development
- **CEC320** — Embedded Systems 1
- **CEC324** — Wireless Technology
- **CEC330** — Computer Networks II
- **COT308** — Technical Writing
- **COT310** — Industrial Experience 2
- **COT312** — Workshop Practice 3
- **EEC330** — Basics of Digital Electronics
- **EEC338** — Probability and Statistics for Engineering
- **CEC417** — Mobile Application Development
- **CEC419** — Design Project
- **CEC427** — Fundamentals of Data Analytics
- **CEC431** — Software Process and Quality
- **CEC441** — TCP/IP Socket Programming
- **CEC443** — Mobile wireless Networks
- **CEC445** — Network Routing algorithms
- **CEC461** — Information Systems Security
- **CEC437** — DevOps and IT Operations Engineering
- **CEC412** — Introduction to Virtual Instrumentation
- **CEC418** — Software Construction and Evolution
- **CEC434** — Data Visualisation
- **CEC442** — Data centric Network



- **CEC444** — Optical Networks
- **CEC446** — Wireless Sensor Networks

Electives Courses for Undergraduate

Option Software Engineering

- **CEC433** — Mobile Applications for Embeded Systems
- **CEC435** — Machine Learning
- **CEC420** — Data Mining
- **CEC424** — Introduction to Deep Learning
- **CEC430** — Full Stack Web Development Technologies
- **CEC436** — Software Engineering for Wireless Applications
- **CEC498** — B.TECH Project

Computer Networks and Systems Maintenance

- **CEC447** — Behavioral Security
- **CEC449** — Cybersecurity principles
- **CEC448** — System administration
- **CEC450** — Enterprise Core Networking
- **CEC452** — Network Design and Implementation

Undergraduate Courses offered to other departments

- **CEC214** — Object Oriented Programming Languages
- **CEC405** — Microprocessor/Microcontroller Practicals
- **CEC328** — Microprocessor/Microcontroller in Mechatronics

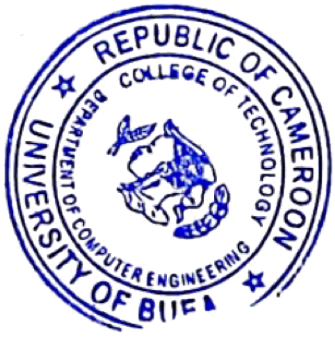
Compulsory Courses for Postgraduate

University Course(s)

- **ENT600** — Entrepreneurship

Departmental Courses

- **CEC601** — Advanced Software Development
- **CEC603** — Information Systems and Network Security
- **CEC605** — Cloud Architecture
- **CEC621** — Artificial Intelligence
- **EEC605** — Probability and Stochastic Processes
- **COT602** — Big Data Analytics
- **CEC610** — Utility Computing
- **CEC612** — FPGA Based Embedded Systems
- **CEC616** — Deep Learning
- **CEC618** — Advanced Data Structures and Algorithms
- **CEC691** — Research Methodology and Scientific Writing
- **ENT600** — Entrepreneurship
- **CEC696** — Pre-defense Seminar
- **CEC698** — M.TECH Dessertation



III. PROGRAM STRUCTURE — OPTION: SOFTWARE ENGINEERING

III-A. Year One — Level 200 (1 Credit = 15 Hours)

TABLE I
CURRICULUM — SOFTWARE ENGINEERING OPTION: YEAR ONE — LEVEL 200 (1 CREDIT = 15 HOURS)

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
CEC209	Introduction to Networking and Security	4	C	40	0	20	60
CEC213	Fundamentals of Programming 1	3	C	20	0	25	45
CEC217	Computer Architecture	4	C	30	10	20	60
CEC221	Static Web Design	3	C	20	0	25	45
CEC223	Algorithm and Complexity	4	C	40	0	20	60
COT201	Workshop Practice 1	3	C	0	0	45	45
EEC209	Mathematics 1	3	C	30	15	0	45
ENG100	Use of English	2	UC	20	10	0	30
FRE100	Functional French	2	UC	20	10	0	30
SPT100	Sports	2	UC	0	0	30	30
TOTAL		**30**		**220**	**45**	**185**	**450**
	SECOND SEMESTER						
CEC218	Introduction to Data Analysis and Machine Learning	4	C	30	10	20	60
CEC220	Principles of Operating Systems and Compilers	4	C	40	0	20	60
CEC224	Software Architecture	4	C	40	20	0	60
CEC226	Script Language	4	C	30	10	20	60
COT210	Industrial Experience 1	2	C	0	0	30	30
CEE100	Civics and Ethics	2	UC	20	10	0	30
EEC204	Mathematics 2	3	C	30	15	0	45
ICT100	Computer for Business	2	UC	15	0	15	30
ENT100	Entrepreneurship	2	UC	20	10	0	30
LAW100	Legal Aspects of Business	2	UC	20	10	0	30
TOTAL		**29**		**245**	**85**	**105**	**435**

III-B. Year Two — Level 300 (1 Credit = 15 Hours)

TABLE II
CURRICULUM — SOFTWARE ENGINEERING OPTION: YEAR TWO — LEVEL 300 (1 CREDIT = 15 HOURS)

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
CEC305	Data Structure and Algorithm	4	C	45	15	0	60
CEC315	Introduction to Cloud Computing	4	C	30	0	30	60
CEC317	Database I	3	C	20	0	25	45
CEC319	Dynamic Web Design	4	C	20	0	40	60
CEC321	Programming with UML	4	C	30	0	30	60
CEC323	Introduction to Artificial Intelligence	4	C	40	0	20	60
COT305	Mathematics 3	3	C	30	15	0	45
COT309	Workshop Practice 2	3	C	0	0	45	45
TOTAL		**29**		**215**	**30**	**190**	**435**
	SECOND SEMESTER						
CEC302	Introduction to Object Oriented Programming	4	C	30	0	30	60
CEC304	Data Security and Integrity	4	C	30	0	30	60
CEC306	System Analysis and Design	4	C	20	10	30	60
CEC318	Introduction to Mobile Application Development	4	C	20	10	30	60
CEC320	Embedded Systems 1	3	C	30	5	10	45
COT308	Technical Writing	2	C	20	10	0	30
COT310	Industrial Experience 2	4	C	0	0	60	60
COT312	Workshop Practice 3	3	C	0	0	45	45
EEC330	Basics of Digital Electronics	3	C	30	15	0	45
EEC338	Probability and Statistics for Engineering	3	C	30	15	0	45
TOTAL		**34**		**210**	**65**	**235**	**510**

REPUBLIC OF CAMEROON
COLLEGE OF TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING
UNIVERSITY OF BUEA

III-C. Year Three — Level 400 (1 Credit = 15 Hours)

TABLE III
CURRICULUM — SOFTWARE ENGINEERING OPTION: YEAR THREE — LEVEL 400 (1 CREDIT = 15 HOURS)

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
CEC417	Mobile Application Development	4	C	40	0	20	60
CEC419	Design Project	6	C	0	0	90	90
CEC427	Fundamentals of Data Analytics	4	C	30	10	20	60
CEC431	Software Process and Quality	4	C	30	0	30	60
CEC461	Information Systems Security	4	C	30	0	30	60
CEC437	DevOps and IT Operations Engineering	4	C	40	20	0	60
CEC4xx	Elective Course	4	E	30	0	30	60
TOTAL		**30**		**200**	**30**	**220**	**450**
	SECOND SEMESTER						
CEC412	Introduction to Virtual Instrumentation	4	C	30	0	30	60
CEC418	Software Construction and Evolution	4	C	30	0	30	60
CEC434	Data Visualisation	4	C	30	0	30	60
CEC498	B.Tech Project	10	C	0	0	150	150
CEC4yy	Elective Course	4	E	30	0	30	60
CEC4zz	Elective Course	4	E	30	0	30	60
TOTAL		**30**		**150**	**0**	**300**	**450**

Elective Courses — Software Engineering

TABLE IV
ELECTIVE COURSES — SOFTWARE ENGINEERING

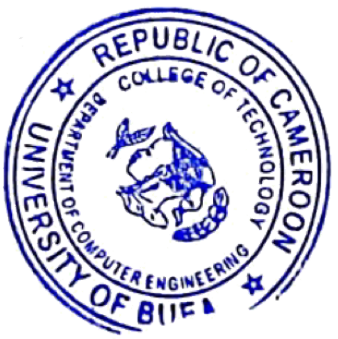
First Semester Electives	Second Semester Electives
	CEC420 Data Mining
CEC433 Mobile Applications for Embedded Systems	CEC424 Introduction to Deep Learning
CEC435 Machine Learning	CEC430 Full Stack Web Development Technologies
	CEC436 Software Engineering for Wireless Applications

Software Engineering — Credit Summary

TABLE V
SOFTWARE ENGINEERING — CREDIT SUMMARY

Parameter	Credits	Lecture Hours	Tutorial Hours	Practical Hours
Year One	59	465	130	290
Year Two	63	425	95	425
Year Three	60	350	30	520
TOTAL	**182**	**1240**	**255**	**1235**

Total Contact Hours: 2730



IV. PROGRAM STRUCTURE — OPTION: COMPUTER NETWORKS

IV-A. Year One — Level 200 (1 Credit = 15 Hours)

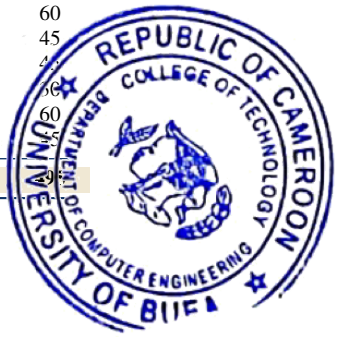
TABLE VI
CURRICULUM — COMPUTER NETWORKS OPTION: YEAR ONE — LEVEL 200 (1 CREDIT = 15 HOURS)

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
CEC209	Introduction to Networking and Security	4	C	40	0	20	60
CEC213	Fundamentals of Programming 1	3	C	20	0	25	45
CEC217	Computer Architecture	4	C	30	10	20	60
CEC221	Static Web Design	3	C	20	0	25	45
CEC223	Algorithm and Complexity	4	C	40	0	20	60
COT201	Workshop Practice 1	3	C	0	0	45	45
EEC209	Mathematics 1	3	C	30	15	0	45
ENG100	Use of English	2	UC	20	10	0	30
FRE100	Functional French	2	UC	20	10	0	30
SPT100	Sports	2	UC	0	0	30	30
TOTAL		**30**		**220**	**45**	**185**	**450**
	SECOND SEMESTER						
CEC218	Introduction to Data Analysis and Machine Learning	4	C	30	10	20	60
CEC220	Principles of Operating Systems and Compilers	4	C	40	0	20	60
CEC224	Software Architecture	4	C	40	20	0	60
CEC226	Script Language	4	C	30	10	20	60
COT210	Industrial Experience 1	2	C	0	0	30	30
CEE100	Civics and Ethics	2	UC	20	10	0	30
EEC204	Mathematics 2	3	C	30	15	0	45
ICT100	Computer for Business	2	UC	15	0	15	30
ENT100	Entrepreneurship	2	UC	20	10	0	30
LAW100	Legal Aspects of Business	2	UC	20	10	0	30
TOTAL		**29**		**245**	**85**	**105**	**435**

IV-B. Year Two — Level 300 (1 Credit = 15 Hours)

TABLE VII
CURRICULUM — COMPUTER NETWORKS OPTION: YEAR TWO — LEVEL 300 (1 CREDIT = 15 HOURS)

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
CEC305	Data Structures and Algorithms	4	C	45	15	0	60
CEC315	Introduction to Cloud Computing	4	C	30	0	30	60
CEC317	Database I	3	C	20	0	25	45
CEC319	Dynamic Web Design	4	C	20	0	40	60
CEC327	Python for Networking	4	C	30	0	30	60
CEC329	Computer Networks I	4	C	40	0	20	60
COT305	Mathematics 3	3	C	30	15	0	45
COT309	Workshop Practice 2	3	C	0	0	45	45
TOTAL		**29**		**215**	**30**	**190**	**435**
	SECOND SEMESTER						
CEC302	Introduction to Object Oriented Programming	4	C	30	0	30	60
CEC304	Data Security and Integrity	4	C	30	0	30	60
CEC320	Embedded Systems 1	3	C	30	0	15	45
CEC324	Wireless Technology	3	C	30	0	15	45
CEC330	Computer Networks II	4	C	20	10	30	60
EEC330	Basics of Digital Electronics	3	C	30	15	0	45
EEC338	Probability and Statistics for Engineering	3	C	30	15	0	45
COT308	Technical Writing	2	C	20	10	0	30
COT310	Industrial Experience 2	4	C	0	0	60	60
COT312	Workshop Practice 3	3	C	0	0	45	45
TOTAL		**33**		**220**	**50**	**225**	**495**



IV-C. Year Three — Level 400 (1 Credit = 15 Hours)

TABLE VIII
CURRICULUM — COMPUTER NETWORKS OPTION: YEAR THREE — LEVEL 400 (1 CREDIT = 15 HOURS)

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
CEC441	TCP/IP Socket Programming	4	C	40	0	20	60
CEC419	Design Project	6	C	0	0	90	90
CEC443	Mobile Wireless Networks	4	C	30	0	30	60
CEC461	Information Systems Security	4	C	30	0	30	60
CEC437	DevOps and IT Operations Engineering	4	C	40	20	0	60
CEC445	Network Routing Algorithms	4	C	30	10	20	60
CEC4xx	Elective Course	4	E	30	0	30	60
TOTAL		**30**		**200**	**30**	**220**	**450**
	SECOND SEMESTER						
CEC442	Data Centric Network	4	C	30	0	30	60
CEC444	Optical Networks	4	C	30	0	30	60
CEC446	Wireless Sensor Networks	4	C	30	0	30	60
CEC498	B.Tech Project	10	C	0	0	150	150
CEC4yy	Elective Course	4	E	30	0	30	60
CEC4zz	Elective Course	4	E	30	0	30	60
TOTAL		**30**		**150**	**0**	**300**	**450**

Elective Courses — Computer Networks

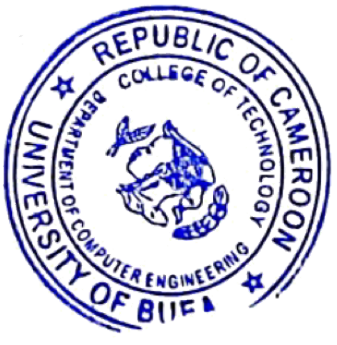
TABLE IX
ELECTIVE COURSES — COMPUTER NETWORKS

First Semester Electives	Second Semester Electives
CEC447 Behavioral Security	CEC448 System Administration
CEC449 Cybersecurity Principles	CEC450 Enterprise Core Networking
	CEC452 Network Design and Implementation

TABLE X
COMPUTER NETWORKS — CREDIT SUMMARY

Parameter	Credits	Lecture Hours	Tutorial Hours	Practical Hours
Year One	59	465	130	290
Year Two	62	435	80	415
Year Three	60	350	30	520
TOTAL	**181**	**1250**	**240**	**1225**

Total Contact Hours: 2715



V. TOP-UP PROGRAMME — REMEDIAL COURSES (HND Students)

V-A. Software Engineering Option

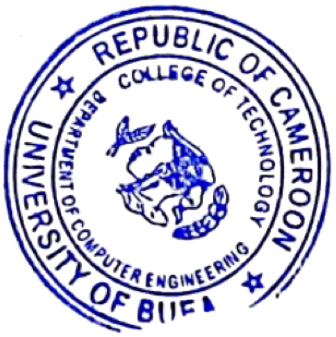
TABLE XI
CURRICULUM: SOFTWARE ENGINEERING OPTION

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
ENG100	Use of English	2	UC	20	10	0	30
FRE100	Functional French	2	UC	20	10	0	30
SPT100	Sports	2	UC	0	0	30	30
CEC315	Introduction to Cloud Computing	4	C	30	0	30	60
CEC321	Programming with UML	4	C	30	0	30	60
TOTAL		**14**		**100**	**20**	**90**	**210**
	SECOND SEMESTER						
ICT100	Computer for Business	2	UC	15	0	15	30
ENT100	Entrepreneurship	2	UC	20	10	0	30
LAW100	Legal Aspects of Business	2	UC	20	10	0	30
CVE100	Civics and Ethics	2	UC	40	20	0	60
CEC304	Data Security and Integrity	4	C	30	0	30	60
CEC318	Introduction to Mobile Application Development	4	C	30	0	15	45
TOTAL		**16**		**155**	**40**	**60**	**255**

V-B. Computer Networks Option

TABLE XII
CURRICULUM: COMPUTER NETWORKS OPTION

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
FRE100	Functional French	2	UC	20	10	0	30
ENG100	Use of English	2	UC	20	10	0	30
SPT100	Sports	2	UC	0	0	30	30
CEC315	Introduction to Cloud Computing	4	C	30	0	30	60
CEC327	Python for Networking	4	C	30	0	30	60
TOTAL		**14**		**100**	**20**	**90**	**210**
	SECOND SEMESTER						
ICT100	Computer for Business	2	UC	15	0	15	30
ENT100	Entrepreneurship	2	UC	20	10	0	30
LAW100	Legal Aspects of Business	2	UC	20	10	0	30
CVE100	Civics and Ethics	2	UC	40	20	0	60
CEC302	Introduction to Object Oriented Programming	4	C	30	0	30	60
CEC304	Data Security and Integrity	4	C	30	0	30	60
TOTAL		**16**		**155**	**40**	**75**	**270**



VI. POSTGRADUATE PROGRAMME — M.TECH IN SOFTWARE ENGINEERING

VI-A. Year Four (1 Credit = 10 Hours)

TABLE XIII
CURRICULUM: YEAR FOUR (1 CREDIT = 10 HOURS)

Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER — 1 Credit = 10 Hours						
CEC601	Advanced Software Development	6	C	40	5	15	60
CEC603	Information Systems and Network Security	6	C	40	5	15	60
CEC605	Cloud Architecture	6	C	40	5	15	60
CEC621	Artificial Intelligence	6	C	40	5	15	60
EEC605	Probability and Stochastic Processes	6	C	45	10	5	60
TOTAL		**30**		**205**	**30**	**65**	**300**
	SECOND SEMESTER						
COT602	Big Data Analytics	6	C	40	5	15	60
CEC610	Utility Computing	6	C	40	5	15	60
CEC612	FPGA Based Embedded Systems	6	C	40	5	15	60
CEC616	Deep Learning	6	C	40	5	15	60
CEC618	Advanced Data Structures and Algorithms	6	C	40	5	15	60
TOTAL		**30**		**200**	**25**	**75**	**300**

VI-B. Year Five (1 Credit = 10 Hours)

TABLE XIV
CURRICULUM: YEAR FIVE (1 CREDIT = 10 HOURS)

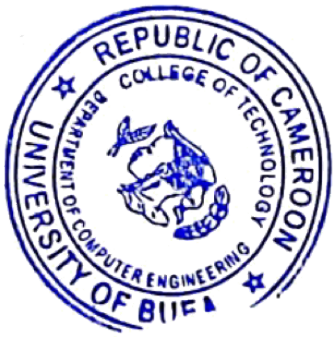
Course Code	Course Title	Credits	Status	L	T	P	Total
	FIRST SEMESTER						
CEC691	Research Methodology and Scientific Writing	6	C	50	10	0	60
ENT600	Entrepreneurship	6	C	50	10	0	60
TOTAL		**12**		**100**	**20**	**0**	**120**
	SECOND SEMESTER						
CEC696	Pre-defense Seminar	6	C	0	0	60	60
CEC698	M.Tech Dissertation	42	C	0	0	420	420
TOTAL		**48**		**0**	**0**	**480**	**480**

Postgraduate — Credit Summary

TABLE XV
POSTGRADUATE — CREDIT SUMMARY

Parameter	Credits	Lecture Hours	Tutorial Hours	Practical Hours
Year Four	60	405	55	140
Year Five	60	100	20	480
TOTAL	**120**	**505**	**75**	**620**

Total Contact Hours: 1200



VII. COURSE DESCRIPTIONS — LEVEL 200

Option: Software Engineering / Computer Networks and Systems Maintenance

VII-1) CEC209 — Introduction to Networking and Security

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Anumoh Michael / Kome Kole
Prerequisites: Introduction to Networks (ITN) and Switching, Routing courses must be completed prior to this course.

Course Description

This course provides students with a thorough understanding of network security, including threats, vulnerabilities, and best practices for protecting networks. Students will configure security measures and troubleshoot network security issues, preparing them for entry-level network and security career opportunities.

Objectives

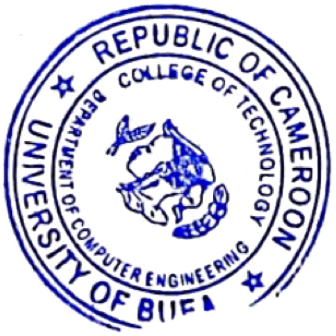
- Provide an in-depth, theoretical understanding of network security
- Enable students to design and support secure network infrastructure
- Prepare students for entry-level jobs in the networking industry

Course Content

- Current network security landscape and evolution of network threats
- Types of threats and attacks; tools used by threat actors
- Reconnaissance, access, and social engineering attacks
- Malware mitigation; IP, TCP, and UDP vulnerability exploitation
- Best practices for protecting networks; common cryptographic processes
- Access Control Lists (ACLs): wildcard masks, standard and extended IPv4 ACLs
- Creating and implementing IPv4 ACLs; securing administrative access
- Configuring IPS to mitigate attacks; testing network security
- Creating a technical security policy; troubleshooting common network security issues

Learning Outcomes

- Describe the current state of cybersecurity and vectors of data loss
- Identify and explain tools used by threat actors to exploit networks
- Describe types of malware and common network attacks
- Explain how IP, TCP, and UDP vulnerabilities are exploited
- Describe best practices for protecting a network
- Describe common cryptographic processes used to protect data in transit
- Explain how ACLs filter network traffic and use wildcard masks
- Configure IPS to mitigate attacks; test network security
- Troubleshoot common network security issues



References & Materials

- Cisco Packet Tracer (software-based lab environment)
- Wireshark — latest stable version
- SSH client software (PuTTY or Tera Term)
- Oracle VirtualBox with Security Workstation VM
- Cisco ASA 5506-X with FirePOWER services

Tools: Ophcrack, L0phtCrack, THC Hydra, RainbowCrack, Medusa, Aircrack-ng, Kismet, InSSIDer, Tcpdump, Ettercap, Wireshark

VII-2) CEC213 — Fundamentals of Programming 1

Semester: 1 **Credits:** 3 **Department:** Computer Engineering **Course Master:** Megoze Hilaire
Prerequisites: None

Course Description

This course introduces students to the fundamental concepts of computer programming and algorithm design. Students will learn to analyze problems, design algorithmic solutions, and implement them using structured programming techniques. The course covers core programming constructs including variables, control structures, loops, functions, and basic data structures.

Objectives

- Describe how to find computerizable solutions to a problem
- Design a well-defined sequence of steps to solve basic problems
- Make efficient use of control structures to handle the flow of programs
- Make use of common data structures to write efficient algorithms
- Explain common searching and sorting algorithms
- Compare two algorithms in terms of correctness and efficiency

Course Content

- Introduction to programming: variables, data types, and declaration
- Structure of a basic program; input/output operations
- Conditional instructions: if-else, switch statements
- Iterative instructions: for, while, do-while loops
- Sub-routines and functions: definition, parameters, return values
- Repetitive instructions and recursion
- Introduction to data structures: arrays, stacks, queues, linked lists, trees
- Searching algorithms: linear search, binary search
- Sorting algorithms: bubble sort, selection sort, insertion sort

Learning Outcomes

- Understand a variety of techniques for designing algorithms
- Understand and appropriately use a wide variety of data structures to solve problems
- Understand and implement fundamental algorithms
- Write, test, and debug basic programs

References & Materials

- Lecture notes
- Kernighan, B.W. & Ritchie, D.M. — The C Programming Language (2nd ed.)
- Sedgewick, R. — Algorithms (4th ed.)

Tools: GCC/G++ compiler, Code:: Blocks IDE, Visual Studio Code

VII-3) CEC217 — Computer Architecture

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Kome Kole
Prerequisites: None

Course Description

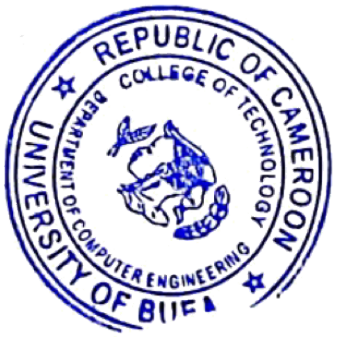
This course provides a comprehensive introduction to the organization and design of modern computer systems. Students will explore the relationship between hardware and software, study instruction set architectures, and understand how processors, memory hierarchies, and I/O systems work together to execute programs.

Objectives

- Understand the fundamental organization of a computer system
- Explore instruction set architectures (ISA) and their design principles
- Understand processor design including pipelining and performance optimization
- Study memory hierarchy: caches, RAM, virtual memory
- Examine I/O systems and peripheral device communication

Course Content

- Computer organization overview: hardware/software interface
- Number systems and data representation: binary, hexadecimal, floating-point
- Instruction Set Architecture (ISA): RISC vs CISC
- Arithmetic and Logic Units (ALU): operations and design
- Control unit design: hardwired and microprogrammed
- Pipelining: hazards and performance improvement
- Memory systems: cache, RAM, virtual memory, paging and segmentation
- Input/Output systems: buses, interrupts, DMA
- Parallel architectures: multi-core processors



Learning Outcomes

- Describe the structure and function of a modern computer system
- Understand and apply number representation and arithmetic operations
- Analyze ISA design trade-offs
- Explain processor pipelining and identify hazards
- Understand memory hierarchy and virtual memory concepts
- Describe I/O mechanisms including interrupts and DMA

References & Materials

- Patterson, D. & Hennessy, J. — Computer Organization and Design (5th ed.)
- Tanenbaum, A. — Structured Computer Organization
- Lecture notes

Tools: MARIE simulator, MIPS assembly emulator, Logisim

VII-4) CEC221 — Static Web Design

Semester: 1 **Credits:** 3 **Department:** Computer Engineering **Course Master:** Dr. Tchapga Tchito Christian
Prerequisites: None

Course Description

This course introduces students to the foundational technologies used to build static websites. Students will learn to create well-structured, visually appealing, and accessible web pages using HTML and CSS, understanding the principles of web standards and responsive design.

Objectives

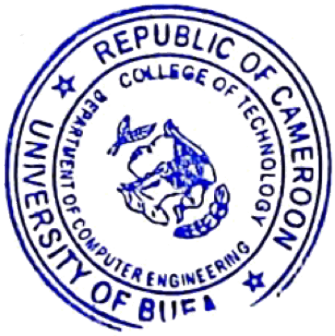
- Understand the structure and purpose of HTML documents
- Apply CSS to style and layout web pages
- Create accessible and standards-compliant web pages
- Implement responsive design for multiple screen sizes

Course Content

- Introduction to the World Wide Web and web technologies
- HTML5: document structure, semantic elements, forms, tables
- CSS3: selectors, box model, layout (Flexbox, Grid)
- Typography, color theory, and visual design principles
- Responsive web design and media queries
- Web accessibility standards (WCAG)
- Introduction to version control with Git
- Web hosting and publishing static sites

Learning Outcomes

- Create well-structured HTML5 documents
- Style web pages using CSS3 with responsive layouts
- Apply web accessibility best practices
- Deploy a static website to a web host



References & Materials

- Duckett, J. — HTML & CSS: Design and Build Websites
- MDN Web Docs (developer.mozilla.org)
- Lecture notes

Tools: VS Code, Git, GitHub Pages, Chrome DevTools, W3C Validator

VII-5) CEC223 — Algorithm and Complexity

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Nyanga Bernard
Prerequisites: CEC213 Fundamentals of Programming 1

Course Description

This course covers the design, analysis, and correctness of algorithms. Students will study fundamental algorithmic paradigms and learn to evaluate algorithm efficiency using Big-O notation. The course bridges theory and practice, equipping students with tools to solve complex computational problems efficiently.

Objectives

- Understand and apply algorithm design paradigms
- Analyze time and space complexity using Big-O notation
- Study classic algorithms for sorting, searching, and graph problems
- Understand NP-completeness and the limits of computation

Course Content

- Introduction to algorithms: correctness and efficiency
- Asymptotic notation: Big-O, Big-Omega, Big-Theta
- Divide and conquer algorithms: merge sort, quicksort, binary search
- Dynamic programming: memoization, tabulation
- Greedy algorithms: Huffman coding, interval scheduling
- Graph algorithms: BFS, DFS, shortest paths (Dijkstra, Bellman-Ford), MST
- String matching algorithms
- Computational complexity: P, NP, NP-complete problems
- Approximation algorithms

Learning Outcomes

- Analyze algorithm correctness using mathematical proof techniques
- Compute time and space complexity of algorithms
- Design efficient algorithms using standard paradigms
- Recognize NP-complete problems and select appropriate strategies

References & Materials

- Cormen, T. et al. — Introduction to Algorithms (CLRS, 3rd ed.)
- Kleinberg, J. & Tardos, E. — Algorithm Design
- Lecture notes

Tools: Python, Java, or C++ (for algorithm implementation), Jupyter Notebook

VII-6) CEC218 — Introduction to Data Analysis and Machine Learning

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Collins Baloko Molimi
Prerequisites: Introduction to AI and Machine Learning, Python programming, Mathematics

Course Description

This course opens the world of data analytics and machine learning using programming. Students will explore foundational concepts in statistics, data processing, and supervised/unsupervised learning, gaining practical skills using industry-standard tools like Jupyter Notebook and Python libraries.

Objectives

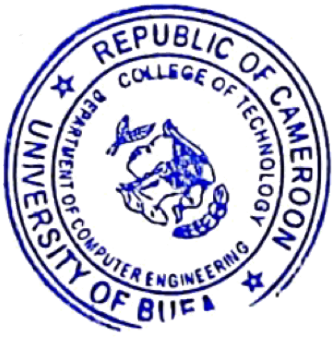
- Understand the fundamentals of data analytics and the data pipeline
- Learn core machine learning concepts and algorithms
- Gain hands-on experience with Python data science libraries
- Apply ML techniques to real-world datasets

Course Content

- Chapter 1: Fundamentals of Data Analytics — introduction, statistics basics, data pipeline
- Chapter 2: Fundamentals of Machine Learning — what is ML, growth, applications
- Chapter 3: Types of Machine Learning — supervised, unsupervised, semi-supervised, reinforcement learning
- Chapter 4: ML and Data Analysis Tasks — classification, clustering, causal discovery, simulation
- Chapter 5: Practical use of ML and data analysis with Python

Learning Outcomes

- Grasp core concepts of data analytics and machine learning
- Understand different types of ML and apply them to real problems
- Use Jupyter Notebook, Anaconda, and Python libraries for analysis
- Execute classification, clustering, and simulation tasks
- Be equipped for junior roles as data analyst or ML engineer



References & Materials

- Hastie, Tibshirani & Friedman — The Elements of Statistical Learning (2nd ed.)
- Murphy — Machine Learning: A Probabilistic Perspective
- James et al. — An Introduction to Statistical Learning
- Alpaydin — Introduction to Machine Learning (4th ed.)
- VanderPlas — Python Data Science Handbook

Tools: Jupyter Notebook, Anaconda, Python (NumPy, Pandas, Scikit-learn, Matplotlib)

VII-7) CEC220 — Principles of Operating Systems and Compilers

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Collins Baloko Molimi
Prerequisites: CEC217 Computer Architecture, CEC213 Fundamentals of Programming 1

Course Description

This course covers the fundamental principles underlying modern operating systems and compiler design. Students will study process management, memory management, file systems, and synchronization in operating systems, as well as the phases of compiler construction from lexical analysis to code generation.

Objectives

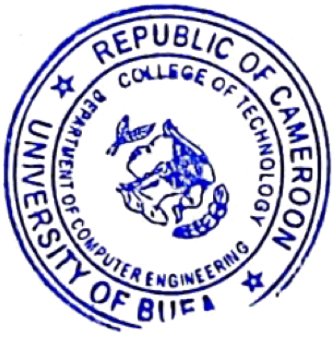
- Understand the role and structure of an operating system
- Study process scheduling, synchronization, and deadlock
- Understand memory management techniques including virtual memory
- Learn the phases of compiler design from source code to machine code

Course Content

- Operating System overview: types, structure, system calls
- Process management: PCB, context switching, scheduling algorithms
- Thread management and concurrency: mutual exclusion, semaphores, monitors
- Deadlock: detection, prevention, avoidance, recovery
- Memory management: paging, segmentation, virtual memory
- File systems: FAT, NTFS, ext4; I/O management
- Introduction to compiler design: phases of compilation
- Lexical analysis: regular expressions and finite automata
- Syntax analysis: context-free grammars, parsing (top-down, bottom-up)
- Semantic analysis, intermediate code generation, optimization, code generation

Learning Outcomes

- Describe the role and structure of an operating system
- Implement and analyze scheduling and synchronization algorithms
- Explain memory management and virtual memory concepts
- Understand the complete pipeline of compiler construction



References & Materials

- Silberschatz, A. et al. — Operating System Concepts (10th ed.)
- Aho, A. et al. — Compilers: Principles, Techniques and Tools (Dragon Book)
- Lecture notes

Tools: Linux/Ubuntu, POSIX threads, Flex & Bison (compiler tools)

VII-8) CEC224 — Software Architecture

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Collins Baloko Molimi
Prerequisites: CEC213 Fundamentals of Programming 1

Course Description

This course introduces students to the principles and practices of software architecture. Students will learn how to design large-scale software systems, evaluate architectural styles and patterns, and apply architectural thinking to real-world software problems. Topics include architectural documentation, quality attributes, and modern architectural approaches.

Objectives

- Understand the role of software architecture in software engineering

- Study and apply major architectural styles and patterns
- Evaluate architectural decisions using quality attributes
- Document software architectures using standard notations

Course Content

- Introduction to software architecture: definitions, role, and importance
- Architectural views and viewpoints: 4+1 view model
- Architectural styles: layered, client-server, pipe-and-filter, event-driven, microservices
- Design patterns: creational, structural, behavioral
- Quality attributes: performance, security, scalability, maintainability
- Architecture evaluation methods: ATAM, SAAM
- Documenting architectures with UML and ADLs
- Cloud-native architectures and containerization
- Case studies: real-world software architectures

Learning Outcomes

- Understand and apply key software architectural styles
- Design software architectures for real-world systems
- Evaluate architecture quality using established methods
- Document software architecture using UML and other notations

References & Materials

- Bass, L. et al. — Software Architecture in Practice (3rd ed.)
- Fowler, M. — Patterns of Enterprise Application Architecture
- Lecture notes

Tools: StarUML, EdrawMax, draw.io, Enterprise Architect

VII-9) CEC226 — Script Language

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Megoze Hilaire
Prerequisites: CEC213 Fundamentals of Programming 1

Course Description

This course introduces students to scripting languages and their use in automating tasks, developing web applications, and processing data. Students will gain practical experience with at least one scripting language (Python and/or Bash) and learn how scripts can automate administrative, data processing, and web development tasks.

Objectives

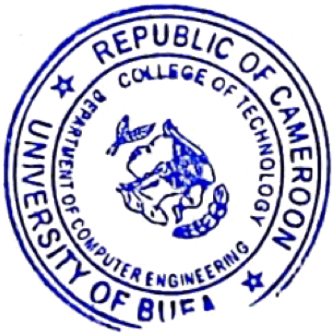
- Understand the role and characteristics of scripting languages
- Write effective scripts for automation and data processing
- Use regular expressions for text processing
- Integrate scripts with operating systems and web technologies

Course Content

- Introduction to scripting: compiled vs interpreted languages
- Python fundamentals: syntax, data types, control flow, functions
- File I/O and regular expressions
- Bash scripting: shell commands, variables, loops, conditionals
- Scripting for system administration and automation
- Web scraping with Python (BeautifulSoup, Requests)
- Working with APIs and JSON data
- Script debugging and error handling

Learning Outcomes

- Write Python and Bash scripts for automation tasks
- Use regular expressions for text manipulation
- Interact with file systems and web APIs from scripts
- Debug and test scripts effectively



References & Materials

- Lutz, M. — Learning Python (5th ed.)
- Sweigart, A. — Automate the Boring Stuff with Python
- Lecture notes

Tools: Python 3, Bash, VS Code, PyCharm

VIII. COURSE DESCRIPTIONS — LEVEL 300

VIII-1) CEC305 — Data Structure and Algorithm

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Nyanga Bernard
Prerequisites: CEC213 Fundamentals of Programming 1, CEC223 Algorithm and Complexity

Course Description

This course provides an in-depth study of fundamental data structures and their associated algorithms. Students will implement and analyze stacks, queues, linked lists, trees, heaps, graphs, and hash tables, learning to choose the most appropriate structure for solving specific computational problems.

Objectives

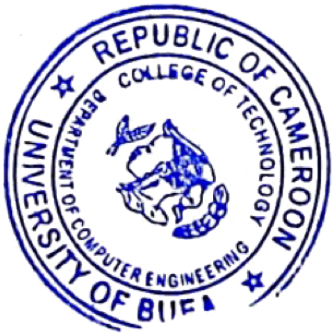
- Implement and analyze fundamental data structures
- Study and apply algorithms on each data structure
- Analyze algorithm complexity and select optimal structures
- Solve practical programming problems using advanced data structures

Course Content

- Review of algorithm analysis: Big-O notation
- Arrays and linked lists: singly, doubly, circular
- Stacks and queues: applications and implementations
- Trees: binary trees, BST, AVL trees, B-trees
- Heaps and priority queues
- Hash tables: hash functions, collision resolution
- Graphs: representation, traversal (BFS/DFS), shortest paths
- Sorting algorithms: quicksort, merge sort, heap sort
- Advanced algorithms: dynamic programming applications

Learning Outcomes

- Implement standard data structures in a programming language
- Analyze time and space complexity of algorithms
- Select appropriate data structures for given problems
- Apply data structures to solve real-world programming problems



References & Materials

- Cormen et al. — Introduction to Algorithms (3rd ed.)
- Sedgewick — Algorithms (4th ed.)
- Lecture notes

Tools: Java, C++, or Python; Eclipse, NetBeans, or VS Code

VIII-2) CEC315 — Introduction to Cloud Computing

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Kometa Denis Mbowoh
Prerequisites: Knowledge of Databases, Computer Fundamentals, Computer Architecture, Networking, and Programming

Course Description

This course introduces students to the concepts, architecture, and services of cloud computing. Students will explore virtualization, cloud service models (IaaS, PaaS, SaaS), cloud security, and hands-on deployment using major cloud platforms such as AWS, Azure, and Google Cloud.

Objectives

- Gain a general overview of cloud computing
- Understand the basic principles and architecture of cloud computing

- Understand cloud service types: IaaS, PaaS, SaaS
- Explore virtualization concepts and set up virtual machines
- Build and deploy applications in a cloud environment
- Understand privacy and security risks associated with cloud computing

Course Content

- Introduction to Cloud Computing
- Benefits of Cloud Computing
- Companies involved in Cloud Computing
- Types of Clouds and Cloud Architectures (public, private, hybrid)
- Virtual Machines and virtualization
- Migrating from standard environment to Cloud
- General Security Concepts in Cloud Computing
- Mini Class Projects

Learning Outcomes

- Understand basic fundamentals of Cloud Computing
- Explain different types of cloud and services offered by providers
- Set up a virtual machine
- Develop and deploy an application in the cloud
- Understand privacy and security issues linked with the Cloud

References & Materials

- Thomas Erl — Cloud Computing: Concepts, Technology & Architecture (Prentice Hall, 2013)
- Anthony T. Velte et al. — Cloud Computing: A Practical Approach (McGraw-Hill, 2010)
- Steve M. Burnett — Amazon Web Services for Beginners (2021)

Tools: VMware, VirtualBox, IDEs, AWS, Microsoft Azure, Google Cloud

VIII-3) CEC317 — Database I

Semester: 1 **Credits:** 3 **Department:** Computer Engineering **Course Master:** Dr. Nkweteyim D. (part-time)
Prerequisites: CEC213 Fundamentals of Programming 1

Course Description

This course introduces the fundamental concepts of relational database management systems. Students will learn to design normalized databases, write SQL queries, and understand transaction management, data integrity, and basic database administration.

Objectives

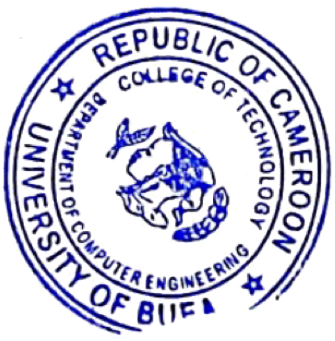
- Understand the relational database model
- Design databases using Entity-Relationship (ER) modeling and normalization
- Write and optimize SQL queries
- Understand transactions, concurrency control, and data integrity

Course Content

- Introduction to databases and DBMS
- Relational model: relations, keys, constraints
- Entity-Relationship (ER) modeling and ER-to-relational mapping
- SQL: DDL (CREATE, ALTER, DROP), DML (SELECT, INSERT, UPDATE, DELETE)
- Joins: inner, outer, self-joins
- Aggregate functions and subqueries
- Normalization: 1NF, 2NF, 3NF, BCNF
- Transactions: ACID properties, concurrency control
- Indexing and performance tuning

Learning Outcomes

- Design a normalized relational database schema
- Write complex SQL queries including joins and subqueries
- Understand and apply transaction management



- Use a DBMS to create and manage databases

References & Materials

- Ramakrishnan, R. & Gehrke, J. — Database Management Systems (3rd ed.)
- Silberschatz et al. — Database System Concepts (7th ed.)
- Lecture notes

Tools: MySQL, PostgreSQL, SQLite, phpMyAdmin, DB Browser for SQLite

VIII-4) CEC319 — Dynamic Web Design

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Tchapga Tchito Christian
Prerequisites: CEC221 Static Web Design, CEC213 Fundamentals of Programming 1

Course Description

Building on the foundations of static web design, this course introduces server-side programming and client-server interactions. Students will build dynamic, data-driven web applications using PHP and/or JavaScript, integrating with databases to create fully functional web systems.

Objectives

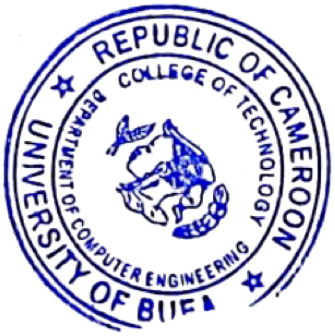
- Understand the client-server model and HTTP protocol
- Develop dynamic web pages using server-side scripting
- Integrate web applications with databases
- Apply JavaScript for client-side interactivity

Course Content

- Review of HTML/CSS and introduction to JavaScript
- JavaScript: DOM manipulation, events, AJAX
- Server-side programming with PHP
- PHP form handling, sessions, and cookies
- Database integration with PHP and MySQL
- Introduction to frameworks (Bootstrap, jQuery)
- Web security basics: input validation, SQL injection prevention
- RESTful API concepts
- Project: Building a complete web application

Learning Outcomes

- Build dynamic, database-driven web applications
- Handle user input securely with server-side validation
- Implement sessions, cookies, and authentication
- Integrate a web front-end with a relational database



References & Materials

- Nixon, R. — Learning PHP, MySQL & JavaScript (5th ed.)
- MDN Web Docs
- Lecture notes

Tools: XAMPP/WAMP, VS Code, PHP, MySQL, JavaScript, Bootstrap

VIII-5) CEC321 — Programming with UML

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Megoze Hilaire
Prerequisites: CEC213 Fundamentals of Programming 1

Course Description

This course introduces the Unified Modeling Language (UML) as a tool for designing and documenting software systems. Students will learn to create various UML diagrams and use them to drive software development, bridging the gap between system design and implementation.

Objectives

- Understand the role of UML in software development
- Create structural and behavioral UML diagrams
- Use UML models to guide code implementation

- Apply model-driven development principles

Course Content

- Introduction to UML: history, purpose, and tool overview
- Use Case Diagrams: actors, use cases, relationships
- Class Diagrams: classes, attributes, methods, associations
- Sequence Diagrams: object interactions over time
- Activity Diagrams: workflow and process modeling
- State Machine Diagrams
- Component and Deployment Diagrams
- Code generation from UML models
- Case study: designing a complete system with UML

Learning Outcomes

- Create and interpret all major UML diagram types
- Use UML to model requirements and system design
- Generate code from UML class diagrams
- Apply UML in the context of a software development project

References & Materials

- Fowler, M. — UML Distilled (3rd ed.)
- Booch, G. et al. — The Unified Modeling Language User Guide
- Lecture notes

Tools: StarUML, EdrawMax, draw.io, Enterprise Architect

VIII-6) CEC323 — Introduction to Artificial Intelligence

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Collins Baloko Molimi / Dr. Tchapga Tchito Christian

Prerequisites: CEC223 Algorithm and Complexity, Mathematics

Course Description

This course introduces the fundamental concepts, techniques, and applications of Artificial Intelligence. Students will explore search algorithms, knowledge representation, machine learning basics, natural language processing, and ethical considerations in AI, gaining a broad foundation for further study in AI and ML.

Objectives

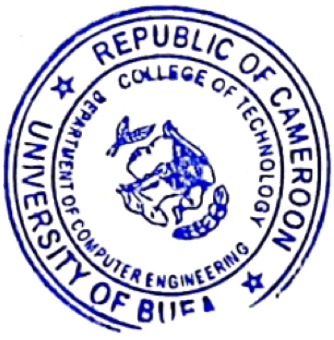
- Understand the history and scope of AI
- Study search algorithms: uninformed and informed search
- Explore knowledge representation and reasoning
- Understand the basics of machine learning and neural networks

Course Content

- Introduction to AI: history, definitions, applications
- Problem solving by search: BFS, DFS, A*, heuristics
- Game playing: minimax and alpha-beta pruning
- Knowledge representation: propositional and first-order logic
- Planning and constraint satisfaction problems
- Introduction to machine learning: supervised, unsupervised
- Neural networks and deep learning overview
- Natural language processing fundamentals
- AI ethics and societal impact

Learning Outcomes

- Implement basic search algorithms for problem solving
- Represent knowledge using logic and apply inference
- Understand foundational machine learning techniques
- Analyze ethical implications of AI systems



References & Materials

- Russell, S. & Norvig, P. — Artificial Intelligence: A Modern Approach (4th ed.)
- Lecture notes

Tools: Python, scikit-learn, Jupyter Notebook, TensorFlow (introduction)

VIII-7) CEC302 — Introduction to Object Oriented Programming

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Nkweteyim D. (part-time)
Prerequisites: CEC213 Fundamentals of Programming 1

Course Description

This course introduces the object-oriented programming (OOP) paradigm as a methodology for designing and building software. Students will learn the core principles of OOP — encapsulation, inheritance, polymorphism, and abstraction — and apply them to build modular, reusable, and maintainable software systems.

Objectives

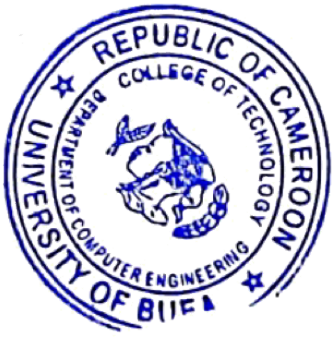
- Understand the principles of object-oriented programming
- Design classes, objects, and class hierarchies
- Apply inheritance, polymorphism, and encapsulation
- Use OOP to build maintainable software systems

Course Content

- Introduction to OOP: classes, objects, methods, attributes
- Encapsulation: access modifiers, getters, setters
- Constructors and destructors
- Inheritance: single and multiple inheritance
- Polymorphism: method overloading and overriding
- Abstraction: abstract classes and interfaces
- Exception handling in OOP
- Collections and generics
- Introduction to design patterns (Singleton, Factory)
- Building a complete OOP project

Learning Outcomes

- Design and implement classes with proper encapsulation
- Apply inheritance and polymorphism in software design
- Handle exceptions in object-oriented programs
- Build a multi-class software project using OOP principles



References & Materials

- Horstmann, C. — Core Java: Fundamentals (11th ed.)
- Eckel, B. — Thinking in Java (4th ed.)
- Lecture notes

Tools: Java or Python, Eclipse/NetBeans/PyCharm, VS Code

VIII-8) CEC304 — Data Security and Integrity

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Anumoh Michael
Prerequisites: CEC209 Introduction to Networking and Security

Course Description

This course examines the principles and techniques used to protect data from unauthorized access, modification, and destruction. Students will study cryptography, authentication mechanisms, database security, and legal/regulatory frameworks governing data protection.

Objectives

- Understand the key principles of information security
- Study cryptographic algorithms and their applications
- Explore authentication and access control mechanisms
- Understand data integrity, digital signatures, and PKI
- Study legal and compliance frameworks for data protection

Course Content

- Information security principles: CIA triad
- Symmetric encryption: DES, AES, RC4
- Asymmetric encryption: RSA, Diffie-Hellman key exchange
- Hash functions: MD5, SHA family
- Digital signatures and certificates
- Public Key Infrastructure (PKI)
- Authentication: passwords, biometrics, MFA
- Access control models: DAC, MAC, RBAC
- Database security: SQL injection prevention, encryption at rest
- Data protection laws: GDPR, Cameroonian data protection regulations

Learning Outcomes

- Apply symmetric and asymmetric encryption techniques
- Implement hashing and digital signatures
- Design access control systems
- Understand and comply with data protection regulations

References & Materials

- Stallings, W. — Cryptography and Network Security (7th ed.)
- Ferguson, N. et al. — Cryptography Engineering
- Lecture notes

Tools: OpenSSL, Python (PyCryptodome), Wireshark

VIII-9) CEC306 — System Analysis and Design

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Nyanga Bernard
Prerequisites: CEC321 Programming with UML

Course Description

This course covers the structured process of analyzing existing systems and designing new ones. Students will apply methodologies including structured analysis, object-oriented analysis, and agile approaches, and produce comprehensive design documentation for software projects.

Objectives

- Understand the systems development life cycle (SDLC)
- Apply structured and object-oriented analysis techniques
- Design system components using appropriate tools and notations
- Produce professional systems documentation

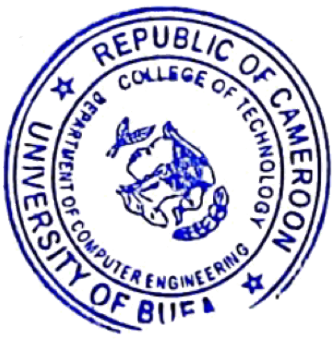
Course Content

- Introduction to systems analysis and design
- Systems Development Life Cycle (SDLC): waterfall, iterative, agile
- Requirements engineering: elicitation, specification, validation
- Use case modeling and system use cases
- Data flow diagrams (DFDs) and data dictionaries
- Object-oriented analysis: CRC cards, class diagrams
- System design: architectural design, interface design
- Database design and ERD
- Implementation and testing planning
- Case study: analyzing and designing a real-world system

Learning Outcomes

- Conduct requirements gathering and produce a requirements specification
- Create DFDs, use case diagrams, and class diagrams
- Design a system architecture and database schema
- Produce a complete system design document

References & Materials



- Dennis, A. et al. — Systems Analysis and Design (6th ed.)
- Whitten, J. & Bentley, L. — Systems Analysis and Design Methods
- Lecture notes

Tools: StarUML, Lucidchart, draw.io, Microsoft Visio

VIII-10) CEC310 — Engineering Ethics and Professional Responsibility

Semester: 2 **Credits:** 3 **Department:** Computer Engineering **Course Master:** Prof. Moffo Lonla Bertrand

Prerequisites: None — compulsory for all Level 300 students (Software Engineering and Computer Networks options)

Course Description

This course introduces the ethical, legal, and professional dimensions of computer engineering practice. Students examine ethical frameworks applicable to computing, the responsibilities of engineers towards society, clients, and the environment, and the regulatory landscape governing software and systems. The course covers health and safety in engineering, sustainable development and green software, AI bias and societal impact, data privacy (GDPR), intellectual property, and professional development. Case studies drawn from Cameroonian and international engineering practice are used throughout.

Objectives

Explain classical ethical theories and apply them to engineering decision-making

Apply the ACM Code of Ethics and IEEE Code of Ethics to professional scenarios

Analyse legal and regulatory requirements including data protection (GDPR), intellectual property, and Cameroonian IT law

Identify health, safety, and environmental responsibilities in engineering projects

Evaluate the societal and ethical implications of AI/ML systems and emerging technologies

Assess software sustainability and green computing principles in engineering design

Demonstrate professional conduct, communication, and responsibility in engineering contexts

Course Content

Unit 1 — Ethical Frameworks: ethical theories (utilitarianism, deontology, virtue ethics); engineering as a profession; ACM and IEEE Codes of Ethics

Unit 2 — Legal and Regulatory Framework: software licensing and IP; data protection and privacy (GDPR, Cameroonian Law No. 2010/012); cybercrime legislation

Unit 3 — Health, Safety and Risk: occupational health and safety in IT environments; risk assessment and management; engineering liability

Unit 4 — Sustainability and Green Computing: environmental impact of computing; energy-efficient software design; e-waste; SDGs and engineering

Unit 5 — AI, Bias and Societal Impact: algorithmic bias and fairness; AI accountability; societal consequences of automation; ethical AI design principles

Unit 6 — Professional Practice: professional development and lifelong learning; technical writing; project and team ethics; entrepreneurship and social responsibility

Learning Outcomes

Distinguish between ethical theories and apply them to engineering dilemmas (Bloom K2)

Apply the ACM/IEEE Codes of Ethics to evaluate professional scenarios (Bloom K3)

Interpret GDPR and intellectual property regulations relevant to software engineering (Bloom K2)

Assess health, safety, and environmental risks in engineering projects (Bloom K4)

Evaluate ethical implications of AI/ML systems deployed in real-world engineering contexts (Bloom K5)

Design software incorporating sustainability and green computing principles (Bloom K4)

Demonstrate professional and ethical communication in engineering reports and presentations (Bloom K3)

Construct a personal professional ethics statement and career development plan (Bloom K5)

References & Materials

ACM Code of Ethics and Professional Conduct (2018) — acm.org/code-of-ethics

IEEE Code of Ethics — ieee.org/about/corporate/governance/p7-8.html

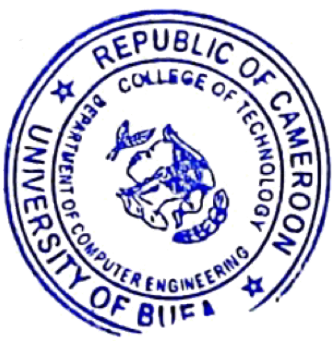
Baase, S. — A Gift of Fire: Social, Legal and Ethical Issues for Computing (4th ed.)

Quinn, M. J. — Ethics for the Information Age (latest edition)

Cameroonian Law No. 2010/012 on Cybersecurity and Cybercrime

General Data Protection Regulation (GDPR) — Official text and ICO guidance

Lecture notes and case studies (provided by Course Master)



VIII-11) CEC318 — Introduction to Mobile Application Development

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Melingui Melono Robert
Prerequisites: CEC302 Introduction to OOP, basic knowledge of Python or Java

Course Description

This course is an introduction to mobile app development targeting both Android and iOS platforms using Flutter. Students will gain skills from app creation through publication on both platforms, covering UI design, state management, and database integration.

Objectives

- Gain basic skills to develop apps for Android and iOS using Flutter
- Understand app design considerations and OS life cycles
- Handle screen orientation, connectivity, and hardware limitations
- Publish an app on mobile platforms

Course Content

- Why Mobile Apps? — transformative devices, reaching customers
- App Design Issues: OS design, Android/iOS life cycles, screen and hardware issues
- Getting started with Flutter: installation, Dart programming language
- Flutter widgets: stateless and stateful, ListView
- Flutter Project Development: building a Simple Interest Calculator and NoteKeeper app
- SQLite database in Flutter using SQFLITE plugin
- Student project: end-to-end mobile app development

Learning Outcomes

- Develop and deploy mobile applications for Android and iOS using Flutter
- Apply UI/UX principles and understand OS differences
- Implement SQLite databases in Flutter apps
- Build and publish a functional app on mobile platforms

References & Materials

- Deitel & Deitel — Mobile Apps with Swift 4 (Pearson)
- Meier, R. — Professional Android (Wrox)
- Flanagan & Eich — JavaScript: The Definitive Guide (6th ed.)

Tools: Android Studio, Flutter, Dart

VIII-12) CEC320 — Embedded Systems 1

Semester: 2 **Credits:** 3 **Department:** Computer Engineering **Course Master:** Tchinda Tandjon Valerry
Prerequisites: CEC217 Computer Architecture, EEC330 Basics of Digital Electronics

Course Description

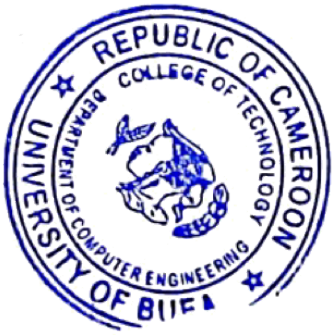
This course introduces the design and programming of embedded systems. Students will explore microcontroller architecture, I/O programming, real-time operating system concepts, and interfacing with peripheral components such as sensors and actuators.

Objectives

- Understand the architecture and components of embedded systems
- Program microcontrollers in C/Assembly
- Interface microcontrollers with sensors, actuators, and communication modules
- Understand real-time constraints in embedded design

Course Content

- Introduction to embedded systems: characteristics and applications
- Microcontroller architecture: Arduino Uno, AVR, ARM Cortex-M
- GPIO programming: digital and analog I/O
- Interrupts, timers, and PWM
- Communication protocols: UART, SPI, I2C
- Interfacing with sensors (temperature, ultrasonic) and actuators (motors, LEDs)
- Introduction to Real-Time Operating Systems (RTOS)



- Power management in embedded systems
- Embedded C programming best practices

Learning Outcomes

- Program a microcontroller to control hardware devices
- Interface sensors and actuators with a microcontroller
- Use communication protocols in embedded systems
- Design a simple embedded application

References & Materials

- Mazidi et al. — AVR Microcontroller and Embedded Systems
- Yiu, J. — The Definitive Guide to ARM Cortex-M3/M4
- Lecture notes

Tools: Arduino IDE, Proteus, Tinkercad, C compiler (avr-gcc)

IX. COURSE DESCRIPTIONS — LEVEL 400

IX-1) CEC417 — Mobile Application Development

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Melingui Melono Robert
Prerequisites: CEC318 Introduction to Mobile Application Development

Course Description

This advanced course builds on introductory mobile development skills. Students will develop professional-grade Android and iOS applications with advanced UI patterns, background processing, network communication, and security, preparing them for industry mobile development roles.

Objectives

- Develop advanced mobile applications using Flutter or native frameworks
- Implement complex UI/UX patterns and animations
- Integrate mobile apps with REST APIs and cloud services
- Apply mobile security best practices

Course Content

- Advanced Flutter: custom widgets, animations, and theming
- State management: Provider, Riverpod, BLoC pattern
- Network communication: HTTP, REST APIs, JSON parsing
- Authentication: OAuth, JWT, Firebase Auth
- Local storage: SharedPreferences, SQLite, Hive
- Push notifications and background services
- Mobile security: certificate pinning, secure storage
- App performance optimization and testing
- Publishing to Google Play Store and Apple App Store
- Capstone project: full-featured mobile application

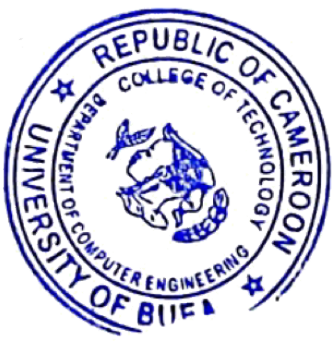
Learning Outcomes

- Build complex, production-ready mobile applications
- Integrate apps with remote APIs and cloud services
- Implement authentication and secure data storage
- Publish an app to major app stores

References & Materials

- Flutter official documentation (flutter.dev)
- Windmill, E. — Flutter in Action
- Lecture notes

Tools: Flutter, Dart, Android Studio, Firebase, Postman, Xcode



IX-2) CEC419 — Design Project

Semester: 1 **Credits:** 6 **Department:** Computer Engineering **Course Master:** All Staff
Prerequisites: Completion of Year 2 requirements

Course Description

The Design Project is a team-based capstone experience where students apply knowledge from across their program to design, develop, and demonstrate a significant software or network engineering project. Students will work in groups under faculty supervision to solve a real-world problem.

Objectives

- Apply integrated knowledge from the Computer Engineering program to a substantial project
- Practice software engineering methodologies in a team environment
- Develop professional communication and project management skills
- Produce a functional prototype and comprehensive documentation

Course Content

- Project planning: scope definition, requirements analysis, timeline
- Team formation and roles: project manager, developers, testers
- Agile development: sprints, stand-ups, retrospectives
- Design phase: system architecture, database design, UI wireframes
- Implementation: coding, code review, version control
- Testing: unit testing, integration testing, user acceptance testing
- Documentation: technical report, user manual
- Presentation and demonstration

Learning Outcomes

- Plan and execute a software engineering project from inception to delivery
- Collaborate effectively as part of a project team
- Produce professional project documentation
- Present and demonstrate a working software product

References & Materials

- Lecture notes
- Project-specific references as assigned

Tools: Git/GitHub, Jira or Trello, project-specific tools (IDE, frameworks, databases)

IX-3) CEC427 — Fundamentals of Data Analytics

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Kometa Denis Mbowoh
Prerequisites: Mathematics (Statistics/Probability), Data Structures, Programming Languages, Database Principles

Course Description

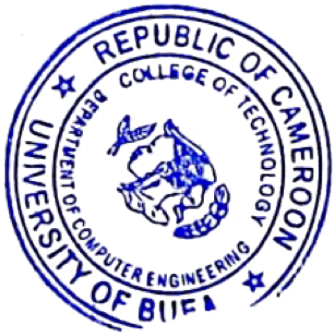
This course introduces students to the full data analytics lifecycle. Students will learn to collect, clean, analyze, and visualize data to derive actionable business insights, using Python, R, and Hadoop as primary tools.

Objectives

- Understand data analytics concepts and the data analytics lifecycle
- Differentiate between BI, data mining, data science, and data analytics
- Learn the step-by-step process of data analytics using Python, R, and Hadoop
- Understand the role of big data in data analytics

Course Content

- Introduction to Data Analysis/Analytics
- Data types, big data principles, and the data analytics lifecycle
- Python packages for data science: NumPy, Pandas, Matplotlib
- Basic insights from datasets: descriptive statistics
- Data preparation: cleaning, wrangling, feature engineering
- ANOVA, correlation, regression, model development and evaluation
- Introduction to R and RStudio; exploratory data analysis
- Introduction to Hadoop and big data frameworks



- Mini class projects and tutorials

Learning Outcomes

- Explain BI, big data, data analytics, and data science concepts
- Perform key tasks in the data analytics lifecycle
- Define business problems and build models for analytics projects
- Use Python and R for data analysis tasks
- Communicate analytics results to non-technical stakeholders

References & Materials

- Moreira et al. — A General Introduction to Data Analytics (Wiley, 2019)
- Maheshwari — Data Analytics Made Accessible (Wiley, 2023)
- Prajapati — Big Data Analytics with R and Hadoop (Packt, 2015)

Tools: Python (NumPy, Pandas, Matplotlib, Scikit-learn), RStudio, Apache Hadoop, Jupyter Notebook

IX-4) CEC431 — Software Process and Quality

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Nyanga Bernard
Prerequisites: CEC306 System Analysis and Design, experience with software development

Course Description

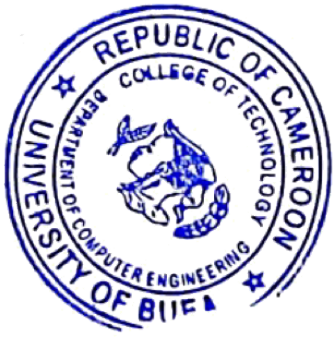
This course covers the principles and practices of software process management and quality assurance. Students will study software development methodologies, measurement techniques, testing strategies, and quality standards used in the software industry.

Objectives

- Understand and compare software development process models
- Apply software quality assurance techniques
- Design and execute comprehensive software testing strategies
- Understand software metrics and how to measure quality

Course Content

- Software process models: waterfall, spiral, RUP, Scrum, XP
- Requirements quality: completeness, consistency, traceability
- Software quality models: ISO/IEC 25010, McCall's model
- Software reviews: inspections, walkthroughs
- Testing fundamentals: unit, integration, system, acceptance testing
- Black-box and white-box testing techniques
- Test automation: JUnit, Selenium, Pytest
- Software metrics: LOC, cyclomatic complexity, code coverage
- Continuous integration and continuous delivery (CI/CD)
- Software quality standards: ISO 9001, CMMI



Learning Outcomes

- Select appropriate software process models for given projects
- Design and execute a test plan for a software system
- Apply software quality metrics to evaluate software products
- Implement CI/CD pipelines for automated testing

References & Materials

- Pressman, R. — Software Engineering: A Practitioner's Approach (8th ed.)
- Myers, G. — The Art of Software Testing (3rd ed.)
- Lecture notes

Tools: JUnit, Selenium, Jenkins, Git, SonarQube

IX-5) CEC461 — Information Systems Security

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Prof. Sone M. (part-time)
Prerequisites: CEC304 Data Security and Integrity, CEC209 Introduction to Networking and Security

Course Description
This course provides a comprehensive study of information systems security management. Students will examine risk management frameworks, security policies, incident response, and compliance requirements in the context of organizational information systems.

Objectives
- Understand information security governance and risk management
- Develop and evaluate security policies and procedures
- Study incident detection, response, and recovery
- Understand compliance frameworks: ISO 27001, NIST, COBIT

Course Content
- Information security governance: strategy, roles, responsibilities
- Risk management: risk identification, assessment, and treatment
- Security policies and procedures
- Physical and environmental security
- Identity and access management (IAM)
- Vulnerability assessment and penetration testing basics
- Incident response and digital forensics fundamentals
- Business continuity and disaster recovery planning
- Security compliance: ISO 27001, NIST Cybersecurity Framework, GDPR

Learning Outcomes
- Develop a risk management framework for an organization
- Write and evaluate security policies
- Plan and execute an incident response procedure
- Understand major security compliance frameworks

References & Materials
- Whitman, M. & Mattord, H. — Principles of Information Security (6th ed.)
- NIST Cybersecurity Framework
- Lecture notes

Tools: Nmap, Nessus (or OpenVAS), Wireshark, Metasploit (educational use)

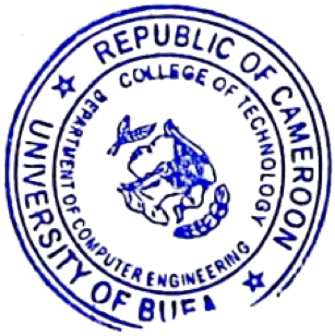
IX-6) CEC412 — Introduction to Virtual Instrumentation

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Prof. Moffo Lonla Bertrand
Prerequisites: CEC217 Computer Architecture, knowledge of programming

Course Description
This course introduces the concept of virtual instrumentation, where software-defined instruments replace traditional hardware instruments. Students will learn to design, develop, and deploy virtual instruments using graphical programming environments, focusing primarily on National Instruments LabVIEW.

Objectives
- Understand the concept and advantages of virtual instrumentation
- Design and develop virtual instruments using LabVIEW
- Acquire, analyze, and visualize data using virtual instruments
- Interface virtual instruments with physical hardware

Course Content
- Introduction to virtual instrumentation and LabVIEW
- LabVIEW environment: front panel, block diagram, palettes
- Data types, structures, and control flow in LabVIEW
- Loops, arrays, and clusters in LabVIEW
- Data acquisition (DAQ): interfacing with sensors and hardware
- Signal processing: filtering, FFT analysis
- File I/O and data logging
- Modular programming: SubVIs and libraries



- Project: building a complete data acquisition and analysis system

Learning Outcomes

- Design and build virtual instruments using LabVIEW
- Acquire real-world data through DAQ hardware
- Process and visualize signals using LabVIEW tools
- Develop modular LabVIEW applications

References & Materials

- Travis, J. & Kring, J. — LabVIEW for Everyone (3rd ed.)
- National Instruments LabVIEW documentation
- Lecture notes

Tools: National Instruments LabVIEW, NI-DAQmx, oscilloscope, signal generators

IX-7) CEC418 — Software Construction and Evolution

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Kometa Denis Mbowoh
Prerequisites: Computing Foundations, Data Structures, Systems Analysis, UML, Software Testing Tools

Course Description

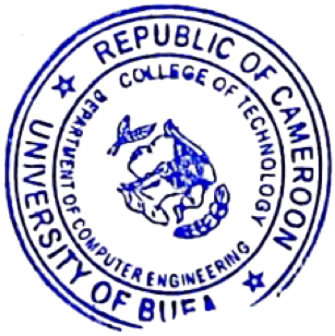
This course covers the principles, methods, and tools for software construction and software evolution. Students will learn the design-to-code cycle, managing complexity and change, and software evolution techniques for maintaining and improving existing systems.

Objectives

- Understand and master software construction principles, methods, and tools
- Learn the design → construct → test cycle
- Manage complexity, change, validation, verification, and standards compliance
- Understand software evolution from the perspective of different activities

Course Content

- Review of Software Development
- Software Construction Fundamentals
- Managing Construction: code quality, refactoring
- Practical Considerations: coding standards, code reviews
- Construction Technologies: defensive programming, error handling
- Software Construction Tools
- Software Evolution: corrective, perfective, adaptive, preventive maintenance
- Reverse engineering and re-engineering
- Case Studies and Tutorials



Learning Outcomes

- Apply the design → construct → test cycle
- Manage complexity and change in software projects
- Apply software evolution techniques
- Use construction tools effectively

References & Materials

- Fairley, R. — Guide to the Software Engineering Body of Knowledge (SWEBOK v3)
- Meyer, B. — Object-Oriented Software Construction (3rd ed.)
- Lecture notes

Tools: StarUML, EdrawMax, VS Code, PyCharm, NetBeans, Eclipse, Python, Java

IX-8) CEC434 — Data Visualisation

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Megoze Hilaire
Prerequisites: CEC427 Fundamentals of Data Analytics, programming knowledge

Course Description

This course covers the principles and techniques for visualizing data to communicate insights effectively. Students will study perception, design principles, and use industry tools to create compelling static and interactive visualizations for a variety of audiences and data types.

Objectives

- Understand the principles of effective data visualization
- Apply visual design and perception principles to data graphics
- Create static and interactive visualizations using Python and web tools
- Tell data stories tailored for different audiences

Course Content

- Introduction to data visualization: history, purpose, and types
- Perception and cognition: how humans interpret visual information
- Design principles: color theory, typography, chart selection
- Static visualization with Python: Matplotlib, Seaborn
- Interactive visualization: Plotly, Bokeh, D3.js basics
- Geospatial data visualization
- Dashboard design and tools: Power BI, Tableau, Dash
- Visualizing time series, networks, and hierarchical data
- Storytelling with data: narrative structures
- Project: data visualization capstone

Learning Outcomes

- Select appropriate chart types for different data and audiences
- Create professional static and interactive visualizations
- Design dashboards for business intelligence
- Tell compelling data stories with visualizations

References & Materials

- Knaflic, C.N. — Storytelling with Data
- Wilke, C. — Fundamentals of Data Visualization
- Murray, S. — Interactive Data Visualization for the Web (D3.js)

Tools: Python (Matplotlib, Seaborn, Plotly), Tableau (or Power BI), D3.js

IX-9) CEC327 — Python for Networking

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Tatsachou Ngningaye D.
Prerequisites: CEC213 Fundamentals of Programming 1, CEC329 Computer Networks I

Course Description

This course focuses on using Python to automate network tasks, interact with network devices, and analyze network data. Students will learn to write scripts that manage network infrastructure, retrieve information from devices, and interact with REST APIs.

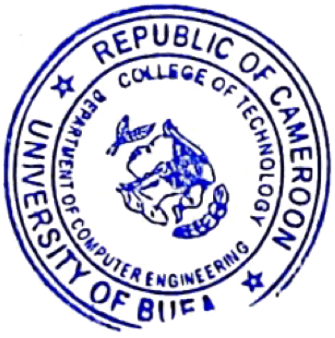
Objectives

- Apply Python programming to networking tasks
- Automate network configuration using Python libraries
- Interact with network device APIs via scripts
- Analyze and process network traffic and log data with Python

Course Content

- Python review: data structures, file I/O, regular expressions
- Network programming fundamentals: sockets, TCP/UDP
- SSH automation with Paramiko and Netmiko
- Working with REST APIs using the Requests library
- Network automation with NAPALM
- Parsing structured data: JSON, XML, YAML
- Jinja2 templates for configuration management
- Interacting with Cisco IOS and NX-OS via Python
- Network data analysis: Pandas and Matplotlib for network data
- Introduction to Ansible as a Python-based automation tool

Learning Outcomes



- Write Python scripts to automate network tasks
- Use Netmiko/Paramiko to connect to and configure network devices
- Parse and generate configuration data using structured formats
- Interact with REST APIs for network management

References & Materials

- Geerling, J. — Ansible for DevOps
- Edelman, J. et al. — Network Programmability and Automation
- Chou, E. — Mastering Python Networking

Tools: Python 3, Netmiko, Paramiko, NAPALM, Ansible, GNS3, Cisco Packet Tracer

IX-10) CEC329 — Computer Networks I

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Anumoh Michael
Prerequisites: Basic computer literacy and internet usage skills

Course Description

This first networking course introduces the architectures, models, protocols, and networking elements that connect users, devices, applications, and data through the internet and across modern computer networks — including IP addressing and Ethernet fundamentals. By the end, students can build simple LANs with IP addressing schemes and basic network security.

Objectives

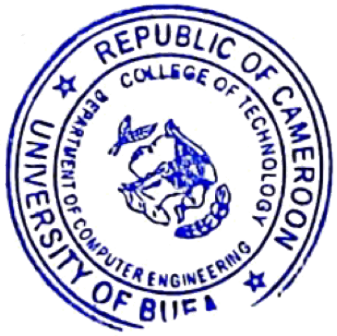
- Understand network architectures, models, and protocols
- Configure switches and routers for basic connectivity
- Implement IPv4 and IPv6 addressing schemes
- Apply basic network security measures

Course Content

- Advances in modern networking: how networks affect daily life
- Network components: hosts, servers, switches, routers
- BYOD, online collaboration, video, and cloud trends
- Basic security threats and solutions
- IP addressing, subnetting, and default gateways
- Cisco IOS: device configuration and command structure
- Physical layer: media types, encoding, signaling
- Data link layer and Ethernet switching
- Number systems: decimal, binary, hexadecimal
- IPv4 subnetting and IPv6 addressing
- Implementing a network design for a small business

Learning Outcomes

- Explain Digital Transformation and its impact
- Configure switches and end devices for local and remote network access
- Configure routers for end-to-end connectivity
- Create IPv4 and IPv6 addressing schemes
- Use security best practices in a small network
- Troubleshoot connectivity issues



References & Materials

- Cisco Packet Tracer (primary lab environment)
- Multimedia learning tools from Cisco NetAcad
- Wireshark, Tera Term

Tools: Cisco Packet Tracer, Wireshark, Tera Term SSH client

IX-11) CEC330 — Computer Networks II

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Tatsachou Ngningaye D.
Prerequisites: CEC329 Computer Networks I

Course Description
Building on Computer Networks I, this course covers advanced switching and routing concepts including VLANs, inter-VLAN routing, dynamic routing protocols, and WAN technologies. Students will configure and troubleshoot more complex network topologies.

Objectives
- Configure VLANs and inter-VLAN routing
- Implement dynamic routing protocols (OSPF, EIGRP)
- Understand WAN technologies and connection methods
- Configure and troubleshoot redundancy protocols (STP, HSRP)

Course Content
- VLAN concepts and configuration
- Inter-VLAN routing: router-on-a-stick, Layer 3 switches
- Spanning Tree Protocol (STP) and its variants (RSTP, MSTP)
- EtherChannel and link aggregation
- Dynamic routing: OSPF single and multi-area, EIGRP
- WAN technologies: leased lines, MPLS, VPN, broadband
- Network Address Translation (NAT) and PAT
- Dynamic Host Configuration Protocol (DHCP)
- First Hop Redundancy Protocols (HSRP, VRRP)
- Network management: SNMP, Syslog, NTP

Learning Outcomes
- Configure VLANs and inter-VLAN routing on Cisco devices
- Implement OSPF and troubleshoot routing issues
- Configure NAT/PAT and DHCP
- Understand and configure STP and redundancy protocols

References & Materials
- Cisco CCNA Official Certification Guide (Wendell Odom)
- Cisco Packet Tracer lab activities
- Lecture notes

Tools: Cisco Packet Tracer, GNS3

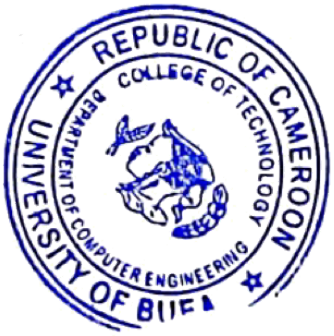
IX-12) CEC324 — Wireless Technology

Semester: 2 **Credits:** 3 **Department:** Computer Engineering **Course Master:** Anumoh Michael
Prerequisites: CEC329 Computer Networks I, Switching and Routing Essentials

Course Description
This course provides a comprehensive introduction to Wireless Local Area Network (WLAN) technology, covering standards, components, operation, channel management, security, and troubleshooting. Students will configure WLANs using both WPA2-PSK and WPA2-Enterprise authentication.

Objectives
- Understand WLAN technology and IEEE 802.11 standards
- Familiarize with WLAN infrastructure components: APs, WLCs
- Understand how CAPWAP is used by WLCs to manage APs
- Identify WLAN threats and implement security mechanisms
- Configure and troubleshoot WLANs

Course Content
- WLAN Technology and Standards: IEEE 802.11a/b/g/n/ac/ax
- Components of a WLAN Infrastructure: APs, WLCs, client devices
- Enabling WLAN Operation: radio frequency, propagation, modulation
- CAPWAP Protocol and WLC Management
- Channel Management: frequency allocation, interference avoidance
- Threats to WLANs and WPA2 security mechanisms



- Configuring WPA2-PSK and WPA2-Enterprise authentication
- Configuring WLC WLAN with VLAN interface and DHCP server
- Troubleshooting wireless configuration issues

Learning Outcomes

- Describe WLAN technology and IEEE 802.11 standards
- Configure a WLAN using WPA2-PSK and WPA2-Enterprise
- Describe how a WLC uses CAPWAP to manage multiple APs
- Troubleshoot common wireless issues

References & Materials

- Coleman & Westcott — CWNA Certified Wireless Network Administrator Study Guide
- Gast, M. — 802.11 Wireless Networks: The Definitive Guide
- Rappaport, T. — Wireless Communications: Principles and Practice

Tools: Cisco Packet Tracer, GNS3, Wireshark, Aircrack-ng, NetSpot, Kali Linux

IX-13) CEC441 — TCP/IP Socket Programming

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Tatsachou Ngningaye D.
Prerequisites: CEC329 Computer Networks I, CEC302 Introduction to OOP

Course Description

This course covers the principles and practice of network programming using the TCP/IP protocol suite and socket APIs. Students will develop client-server applications, implement common network protocols, and learn to build reliable and scalable networked systems.

Objectives

- Understand the TCP/IP protocol suite and the socket API
- Develop client-server applications using TCP and UDP sockets
- Implement network communication protocols
- Handle concurrency in network applications

Course Content

- TCP/IP review: IP, TCP, UDP, ICMP
- The Socket API: creating, binding, connecting, listening sockets
- TCP client-server programming: connection lifecycle
- UDP client-server programming: datagrams
- I/O multiplexing: select(), poll(), epoll()
- Multi-threaded and multi-process network servers
- Non-blocking I/O and asynchronous programming
- Implementing application protocols: HTTP, FTP, SMTP
- Network security in socket programming: TLS/SSL
- Project: building a complete networked application

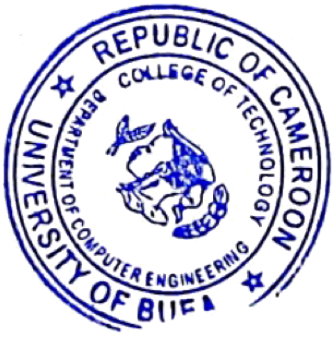
Learning Outcomes

- Write TCP and UDP client-server applications
- Handle concurrent connections in network servers
- Implement common application-layer protocols
- Secure network communications using TLS/SSL

References & Materials

- Stevens et al. — Unix Network Programming Vol. 1 (3rd ed.)
- Donahoo & Calvert — TCP/IP Sockets in C (2nd ed.)
- Lecture notes

Tools: C/Python, Wireshark, Linux networking tools (netstat, tcpdump), OpenSSL



IX-14) CEC443 — Mobile Wireless Networks

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Kome Kole
Prerequisites: CEC329 Computer Networks I, CEC324 Wireless Technology

Course Description

This course examines the architecture, protocols, and standards of mobile wireless networks including cellular systems (2G-5G), mobile data networks, and emerging technologies. Students will study mobility management, handoff mechanisms, and QoS in mobile environments.

Objectives

- Understand the evolution of cellular network technologies (2G to 5G)
- Study mobile network architecture and protocols
- Understand mobility management: handoff, location, roaming
- Explore QoS requirements in mobile networks

Course Content

- Introduction to mobile wireless communications
- Cellular network fundamentals: frequency reuse, cell planning
- GSM (2G): architecture, protocols, authentication
- GPRS/EDGE: data over GSM networks
- UMTS/WCDMA (3G): architecture and radio interface
- LTE (4G): OFDM, MIMO, EPC architecture
- 5G NR: use cases, network slicing, massive MIMO, mmWave
- Mobility management: handover, paging, roaming
- Mobile IP and mobility protocols
- QoS in mobile networks: latency, throughput, packet loss

Learning Outcomes

- Describe the evolution and architecture of cellular networks
- Understand handoff and mobility management mechanisms
- Analyze QoS parameters in mobile environments
- Understand 5G use cases and architecture

References & Materials

- Rappaport, T. — Wireless Communications: Principles and Practice (2nd ed.)
- Dahlman, E. et al. — 5G NR: The Next Generation Wireless Access Technology
- Lecture notes

Tools: NS3 network simulator, MATLAB (for link budget calculations), Wireshark

IX-15) CEC445 — Network Routing Algorithms

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Anumoh Michael
Prerequisites: CEC329 Computer Networks I; basic network OS navigation skills

Course Description

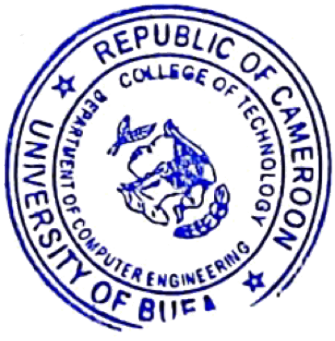
This course focuses on routing technologies and router operations supporting small-to-medium business networks. Students gain a comprehensive understanding of router configuration, routing protocols, and troubleshooting in IPv4 and IPv6 networks.

Objectives

- Understand router operations and packet forwarding decisions
- Configure static and dynamic routing in IPv4 and IPv6 networks
- Understand and configure routing protocols
- Troubleshoot routing configurations

Course Content

- Overview of routers and their roles in networks
- Packet forwarding: routing table lookup process
- Basic router configuration
- Routing table structure and modification



- Static vs dynamic routing concepts
- IPv4 and IPv6 static routes, default static routes
- Floating static routes for backup connections
- IPv4 and IPv6 static host routes
- Troubleshooting static and default route configurations
- Introduction to dynamic routing protocols (OSPF overview)

Learning Outcomes

- Explain how routers use packet information for forwarding
- Configure basic router settings
- Configure IPv4 and IPv6 static and default routes
- Configure floating static routes for redundancy
- Troubleshoot static route configuration issues

References & Materials

- Cisco CCNA Official Certification Guide
- Cisco Packet Tracer labs (31 activities)
- 14 hands-on labs using Cisco 4221/2960 or equivalent

Tools: Cisco Packet Tracer, GNS3, Cisco IOS simulators

IX-16) CEC442 — Data Centric Network

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Anumoh Michael
Prerequisites: Basic Networking Concepts, Computer Hardware fundamentals, Basic Electrical Knowledge, IT Security Awareness

Course Description

This course provides a comprehensive overview of data center infrastructure, management, and operations. Students will explore power systems, cooling technologies, fire protection, rack layout, maintenance strategies, and safety in a data center environment.

Objectives

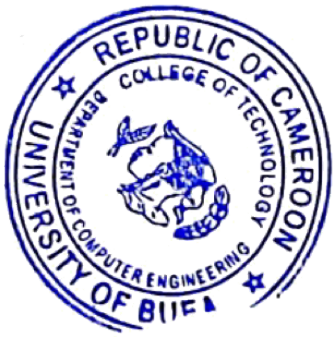
- Understand fundamentals of data center infrastructure and electricity
- Explain the role of UPS systems in power management
- Identify cooling requirements and architectures
- Describe transient temperature management
- Understand fire protection, maintenance, and reliability strategies
- Define data center availability and efficiency metrics

Course Content

- Module 1: Introduction to Data Centers — overview, key components
- Module 2: Electrical Power Systems — batteries, UPS systems, standards
- Module 3: Cooling and Environmental Control — cooling architectures
- Module 4: Rack Layout and Physical Considerations
- Module 5: Fire Protection and Safety Measures
- Module 6: Data Center Maintenance and Reliability
- Module 7: Data Center Availability and Efficiency
- Module 8: Data Center Management and Change Control
- Module 9: Regulatory Standards and Compliance
- Module 10: Case Studies and Practical Exercises

Learning Outcomes

- Describe the fundamental components of a data center
- Explain UPS battery systems and power backup solutions
- Identify cooling architectures and their importance
- Analyze rack layout and airflow management
- Implement preventative and predictive maintenance strategies
- Understand environmental health, safety regulations, and industry standards



References & Materials

- Geng, H. — Data Center Handbook
- Asgarian, V. — Data Center Essentials: Design, Construction, and Operation
- Curtis, P.M. — Maintaining Mission Critical Systems in a 24/7 Environment

Tools: AutoCAD, Cisco Packet Tracer, DCIM software, Schneider EcoStruxure, Nagios/Zabbix

IX-17) CEC444 — Optical Networks

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Anumoh Michael
Prerequisites: CEC329 Computer Networks I, Switching and Routing Essentials

Course Description

This course examines the principles, technologies, and design of optical networks. Topics include fiber optic transmission, WDM systems, optical amplifiers, SONET/SDH, and modern coherent optical systems. Additionally, the course integrates AI and ML applications in networking automation, security, and optimization.

Objectives

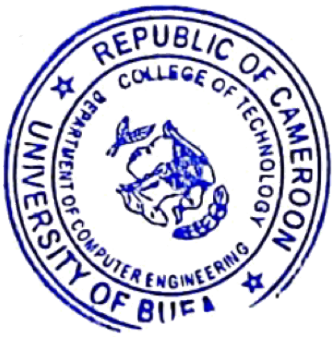
- Understand fiber optic transmission principles
- Study WDM and DWDM systems and components
- Understand SONET/SDH and OTN standards
- Explore AI and ML applications in network automation
- Study SDN, IBN, and AI-powered network management

Course Content

- Introduction to Optical Networking: fiber types, propagation
- Wavelength Division Multiplexing (WDM) and DWDM
- Optical amplifiers: EDFAs, Raman amplifiers
- SONET/SDH: frame structures, multiplexing
- Optical Transport Network (OTN)
- Coherent optical systems and modulation formats
- AI and ML in Networking: fundamentals and applications
- AI-powered network automation, SDN, Intent-Based Networking (IBN)
- AI in Network Security: threat detection, zero trust
- Practical: using ChatGPT as a network simulator; AI-based troubleshooting

Learning Outcomes

- Describe optical fiber transmission principles and components
- Explain WDM systems and their design considerations
- Understand SONET/SDH and OTN standards
- Apply AI and ML concepts to network management
- Configure and troubleshoot network devices using AI-driven tools



References & Materials

- Ramaswami, R. et al. — Optical Networks: A Practical Perspective (3rd ed.)
- Garrett, J. & Ha, A. — AI for Network Engineers
- Edelman et al. — Network Programmability and Automation

Tools: Python, TensorFlow, Cisco Packet Tracer, GNS3, Wireshark, ChatGPT API

IX-18) CEC446 — Wireless Sensor Networks

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Tatsachou Ngningaye D.
Prerequisites: CEC324 Wireless Technology, CEC320 Embedded Systems 1

Course Description

This course covers the design, deployment, and programming of wireless sensor networks (WSNs). Students will study sensor node hardware, energy-efficient communication protocols, data aggregation, and IoT applications of WSNs.

Objectives

- Understand the architecture of wireless sensor networks
- Study energy-efficient MAC and routing protocols for WSNs

- Program sensor nodes for data collection and aggregation
- Apply WSN concepts to IoT applications

Course Content

- Introduction to WSNs: characteristics, challenges, applications
- Sensor node hardware: microcontrollers, sensors, transceivers
- Energy harvesting and power management
- MAC protocols for WSNs: S-MAC, T-MAC, TDMA-based
- Routing protocols: flooding, LEACH, PEGASIS, geographic routing
- Data aggregation and in-network processing
- Localization and time synchronization
- WSN security: eavesdropping, DoS, key management
- IoT integration: MQTT, CoAP, integration with cloud platforms
- Project: deploying a WSN for environmental monitoring

Learning Outcomes

- Design and deploy a wireless sensor network
- Implement energy-efficient communication protocols
- Program sensor nodes using embedded C or Python
- Integrate WSN data with cloud platforms via IoT protocols

References & Materials

- Akyildiz, I.F. & Vuran, M. — Wireless Sensor Networks (Wiley)
- Dargie, W. & Poellabauer, C. — Fundamentals of Wireless Sensor Networks
- Lecture notes

Tools: Arduino, Raspberry Pi, Zigbee/XBee modules, COOJA simulator, MQTT broker

IX-19) CEC452 — Network Design and Implementation

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Anumoh Michael
Prerequisites: CEC329 Computer Networks I, CEC330 Computer Networks II

Course Description

This course covers the principles and methodologies for designing, planning, and implementing enterprise networks. Students will apply structured design frameworks, address scalability and redundancy requirements, and document network designs following industry standards.

Objectives

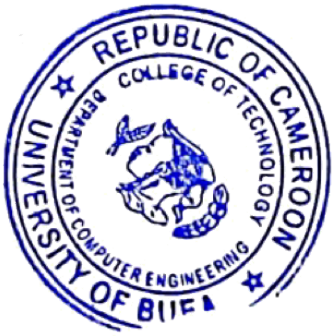
- Apply structured methodologies (PPDIOO) for network design
- Design scalable and highly available network architectures
- Select appropriate hardware and protocols for enterprise requirements
- Document network designs and present proposals

Course Content

- Network design fundamentals and PPDIOO lifecycle
- Hierarchical network design: core, distribution, access layers
- LAN design: VLAN segmentation, link aggregation, redundancy (STP)
- WAN design: MPLS, SD-WAN, VPN technologies
- IP addressing and summarization planning
- Routing protocol selection and design
- Network security design: DMZ, firewalls, IDS/IPS placement
- Wireless network design: site survey, AP placement
- High availability: redundancy, failover, load balancing
- Network documentation: topology diagrams, IP plans, runbooks
- Project: complete network design proposal for a given scenario

Learning Outcomes

- Apply the PPDIOO methodology to a network design project
- Design a hierarchical enterprise network with redundancy



- Select and justify routing protocols for enterprise use
- Produce professional network design documentation

References & Materials

- Oppenheimer, P. — Top-Down Network Design (3rd ed.)
- Cisco Design Guides (cisco.com)
- Lecture notes

Tools: Cisco Packet Tracer, GNS3, Lucidchart, draw.io, Microsoft Visio

X. ELECTIVE COURSE DESCRIPTIONS

X-1) CEC420 — Data Mining

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Kometa Denis Mbowoh
Prerequisites: Mathematics/Statistics, Data Structures, Data Analytics, Python, Database Principles

Course Description

This course covers the principles and techniques of extracting useful knowledge from large datasets. Students will study association rule mining, classification, clustering, and anomaly detection, and learn to use data mining tools and apply them to business and scientific problems.

Objectives

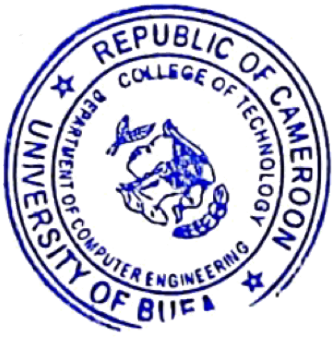
- Understand concepts of data/information and data mining
- Learn association rule mining, classification, and clustering algorithms
- Understand the data mining process and knowledge discovery
- Apply data mining tools to real-world problems

Course Content

- Introduction to Data Mining Principles
- Data Mining and Knowledge Discovery (KDD)
- Data Warehousing and Data Mining overview
- Data Mining Tasks: clustering, classification, rule learning
- Data Mining Process
- Association Rule Mining: Apriori algorithm
- Classification and Prediction: decision trees, Naive Bayes, SVM
- Clustering: k-means, hierarchical, DBSCAN
- Anomaly Detection
- Mini Class Projects and General Tutorial

Learning Outcomes

- Compare functional differences between data warehouses and databases
- Pre-process data and apply mining techniques
- Identify association rules, classification, and clusters in large datasets
- Solve real-world problems using data mining techniques
- Build and deploy Machine Learning models for data mining problems



References & Materials

- Han, J. et al. — Data Mining: Concepts and Techniques (3rd ed.)
- Tan, P. et al. — Introduction to Data Mining (Pearson, 2006)
- Pujari, A. — Data Mining Techniques (3rd ed.)

Tools: Python (scikit-learn, Pandas), WEKA, RapidMiner, Orange, KNIME, Jupyter Notebook

X-2) CEC424 — Introduction to Deep Learning

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Tchapga Tchito Christian
Prerequisites: Introduction to AI and Machine Learning, Python programming, Mathematics

Course Description

This course enables students to understand the applications and concepts of Deep Learning. Students will explore neural network architectures including CNNs, RNNs, and autoencoders, and gain hands-on experience with major deep learning frameworks.

Objectives

- Understand the fundamentals of deep learning
- Explore different neural network architectures
- Apply mathematical foundations of deep learning
- Use deep learning platforms and libraries

Course Content

- Chapter 1: Introduction to Deep Learning — neural networks, vanishing gradient
- Chapter 2: Deep Learning Models — RBMs, Deep Belief Nets, CNNs, RNNs
- Chapter 3: Additional Models — Autoencoders, Recursive Neural Tensor Nets
- Chapter 4: Mathematical Concepts of CNN — convolution, pooling, ReLU, dropout, loss functions
- Chapter 5: Deep Learning Platforms & Libraries — H2O.ai, Theano, Caffe, TensorFlow/Keras, PyTorch
- Chapter 6: Practical Examples and Student Projects

Learning Outcomes

- Define deep learning and explore different neural network architectures
- Comprehend key mathematical concepts: convolution, activation functions, optimization
- Gain hands-on experience with TensorFlow, Keras, and PyTorch
- Implement deep learning projects for real-world AI problems

References & Materials

- Goodfellow, I. et al. — Deep Learning (MIT Press, 2016)
- Chollet, F. — Deep Learning with Python (Manning, 2018)
- Murphy, K.P. — Machine Learning: A Probabilistic Perspective (MIT Press)

Tools: Anaconda, Jupyter Notebook, TensorFlow, Keras, PyTorch, Visual Studio

X-3) CEC430 — Full Stack Web Development Technologies

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Tchapga Tchito Christian / Dr. Melingui Melono Robert

Prerequisites: CEC319 Dynamic Web Design, CEC302 Introduction to OOP

Course Description

This course provides a comprehensive introduction to full-stack web development. Students will build complete web applications from the front-end to the back-end, integrating databases and deploying to cloud platforms using modern frameworks and tools.

Objectives

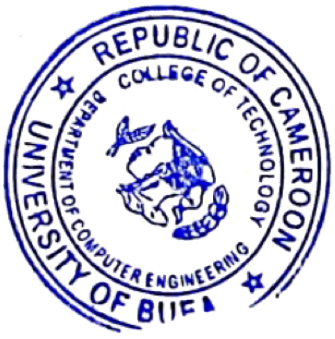
- Develop proficiency in front-end technologies (React or Vue)
- Build RESTful back-end services using Node.js or Django
- Integrate front-end and back-end with a database
- Deploy full-stack applications to cloud platforms

Course Content

- Front-end fundamentals review: HTML5, CSS3, JavaScript ES6+
- Introduction to React.js: components, props, state, hooks
- React Router and SPA navigation
- Back-end development with Node.js and Express.js
- RESTful API design and implementation
- Database integration: MongoDB (NoSQL) and PostgreSQL (SQL)
- Authentication and authorization: JWT, OAuth
- State management: Redux or Context API
- Testing: Jest, React Testing Library
- Deployment: Docker, AWS, Heroku

Learning Outcomes

- Build responsive single-page applications using React
- Develop RESTful APIs with Node.js/Express
- Integrate front-end and back-end with a database



- Deploy full-stack applications to production environments

References & Materials

- Eloquent JavaScript — Marijn Haverbeke
- Node.js official documentation
- React official documentation

Tools: VS Code, Node.js, React.js, MongoDB Atlas, PostgreSQL, Git, Docker, Heroku/AWS

X-4) CEC433 — Mobile Applications for Embedded Systems

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Dr. Melingui Melono Robert
Prerequisites: C/C++ language programming, Object Oriented Programming, overview of mobile app development

Course Description

This course demonstrates how to monitor and control embedded systems using mobile applications on Android/iOS and native operating systems. Students will build mobile apps that interface with microcontrollers and IoT devices.

Objectives

- Show how to monitor embedded systems using mobile applications
- Develop mobile apps for controlling IoT devices
- Interface mobile applications with hardware (Arduino, Raspberry Pi)
- Apply mobile development for smart home and IoT use cases

Course Content

- Overview of Embedded Systems (ES)
- Mobile App Overview: iOS, Android, cross-platform
- Embedded Equipment: Arduino Uno, Raspberry Pi, sensors, actuators
- Mobile app programming to manage Embedded Systems: Smart Home case study
- Bluetooth and Wi-Fi communication between mobile app and embedded system
- Mobile Development App for the Internet of Things (IoT)
- Student projects: Traffic lights, Banking ATM through mobile app

Learning Outcomes

- Understand basic fundamentals of mobile app development for embedded systems
- Achieve simple projects using microcontrollers, sensors, and actuators
- Develop simple mobile apps for controlling embedded systems

References & Materials

- Adiono et al. — Efficient Android Software Development Using MIT App Inventor 2 for Bluetooth-Based Smart Home (Springer, 2019)
- Ibrahim, D. — MIT App Inventor Projects: 50+ Android and iOS Apps with Raspberry Pi (Elektor, 2020)

Tools: MIT App Inventor, Flutter, Android Studio, Arduino IDE, Arduino Uno, Raspberry Pi

X-5) CEC435 — Machine Learning

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Collins Baloko Molimi
Prerequisites: CEC323 Introduction to AI, CEC218 Introduction to Data Analysis and ML, Mathematics/Statistics

Course Description

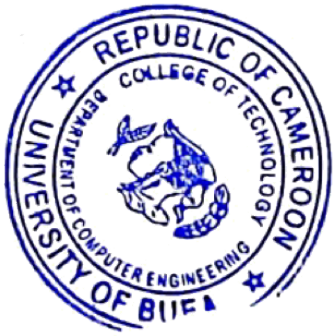
This course provides an in-depth study of machine learning algorithms and their applications. Students will implement supervised, unsupervised, and reinforcement learning algorithms, apply model selection and evaluation techniques, and develop practical ML solutions to real-world problems.

Objectives

- Understand the mathematical foundations of ML algorithms
- Implement and evaluate supervised learning algorithms
- Study unsupervised learning: clustering and dimensionality reduction
- Apply reinforcement learning concepts
- Use scikit-learn and other ML frameworks in practice

Course Content

- Review of probability, statistics, and linear algebra for ML



- Linear regression and gradient descent
- Logistic regression and classification metrics
- Decision trees, Random Forests, and Gradient Boosting
- Support Vector Machines (SVM)
- Neural networks and backpropagation
- Unsupervised learning: k-means, hierarchical clustering, PCA
- Model selection: cross-validation, bias-variance trade-off
- Hyperparameter tuning: Grid Search, Random Search
- Introduction to Reinforcement Learning
- Final project: end-to-end ML pipeline on a real dataset

Learning Outcomes

- Implement key supervised and unsupervised ML algorithms
- Evaluate and select ML models using appropriate metrics
- Apply feature engineering and preprocessing techniques
- Deploy a machine learning pipeline end-to-end

References & Materials

- Géron, A. — Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (3rd ed.)
- Bishop, C. — Pattern Recognition and Machine Learning

Tools: Python, scikit-learn, Pandas, NumPy, Matplotlib, Jupyter Notebook

X-6) CEC436 — Software Engineering for Wireless Applications

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Megoze Hilaire
Prerequisites: CEC302 Introduction to OOP, CEC324 Wireless Technology, CEC318 Introduction to Mobile App Development

Course Description

This course addresses the specific challenges of developing software for wireless and mobile environments. Students will study wireless software architectures, energy-aware programming, adaptive applications, and testing strategies for unreliable network conditions.

Objectives

- Understand the unique challenges of software engineering for wireless environments
- Design wireless applications considering energy constraints
- Apply adaptive and context-aware programming techniques
- Test and validate software for variable network conditions

Course Content

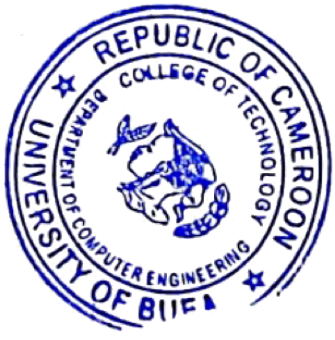
- Introduction to wireless software engineering challenges
- Wireless communication overview: Bluetooth, Zigbee, Wi-Fi, LTE/5G
- Software architectures for wireless systems
- Energy-efficient software design patterns
- Context-aware and adaptive application design
- Mobile middleware and platform APIs
- QoS management in wireless applications
- Testing strategies for wireless environments: simulation, emulation
- Security in wireless software development
- Case studies: IoT, healthcare, and smart city applications

Learning Outcomes

- Design software architectures suitable for wireless environments
- Apply energy-efficient programming techniques
- Test wireless applications under simulated network conditions
- Implement security measures in wireless software

References & Materials

- Holzmann, C. — Mobile Application Development (lecture notes)



- Schiller, J. — Mobile Communications (2nd ed.)
- Lecture notes

Tools: Android Studio, Flutter, NS3, Wireshark, Kali Linux

X-7) CEC437 — DevOps and IT Operations Engineering

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Kometa Denis
Prerequisites: CEC327 Python for Networking, CEC448 System Administration

Course Description

This course introduces students to DevOps culture and IT Operations Engineering. Students gain hands-on experience building CI/CD pipelines, containerising applications with Docker and Kubernetes, provisioning infrastructure with Ansible and Terraform, and monitoring live systems with Prometheus and Grafana. The course bridges software development and operations, preparing graduates for modern ICT engineering roles.

Objectives

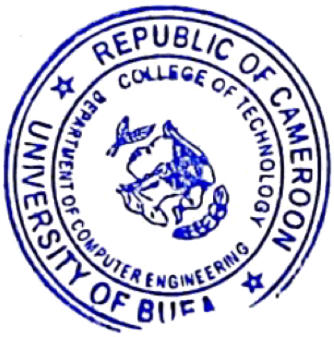
- Explain DevOps principles (CI/CD, IaC, monitoring) and justify their role in modern software delivery
- Design and implement a CI/CD pipeline using Git, Jenkins or GitHub Actions, and Docker
- Build, deploy, and orchestrate containerised applications using Docker and Kubernetes
- Provision and configure servers automatically using Ansible playbooks and Terraform
- Set up monitoring dashboards and alerting with Prometheus and Grafana

Course Content

- DevOps fundamentals: CALMS framework, CI/CD concepts, toolchain overview
- Version control workflows: Git branching strategies, pull requests, code review
- CI/CD pipelines: Jenkins declarative pipelines, GitHub Actions workflows, automated testing
- Containerisation: Docker images, Dockerfile, Docker Compose, container registries and tagging
- Container orchestration: Kubernetes architecture, pods, deployments, services, Helm
- Infrastructure as Code: Ansible playbooks and roles, Terraform providers and state
- Monitoring and observability: Prometheus metrics, Grafana dashboards, ELK stack basics
- DevSecOps : SAST, dependency and container image scanning, secrets management
- Practical mini-project: end-to-end pipeline — code → CI → Docker image → Kubernetes deploy → Prometheus monitoring
- Agile/Scrum in a DevOps context: sprint planning, retrospectives, kanban boards

Learning Outcomes

- Build a CI/CD pipeline that automatically tests, builds, and packages an application
- Containerise a multi-tier application and deploy it to a Kubernetes cluster
- Write an Ansible playbook to idempotently provision a Linux server
- Configure Prometheus alerting and build a Grafana dashboard for a running service



References & Materials

- Kim G. et al. — The DevOps Handbook (2nd ed.), IT Revolution, 2021
- Official documentation: Docker, Kubernetes, Ansible, Terraform, Prometheus, Jenkins
- Lecture notes and lab sheets

Tools: Git, Jenkins, GitHub Actions, Docker, Kubernetes (Minikube), Ansible, Terraform, Prometheus, Grafana, VS Code

X-8) CEC447 — Behavioral Security

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Tatsachou Ngningaye D.
Prerequisites: CEC461 Information Systems Security

Course Description

This course examines the human and behavioral dimensions of cybersecurity. Students will study social engineering, insider threats, security culture, and psychological factors that influence security decisions, along with technical and organizational countermeasures.

Objectives

- Understand the role of human behavior in cybersecurity incidents
- Study social engineering tactics and countermeasures

- Analyze insider threat models and mitigation strategies
- Design security awareness programs for organizations

Course Content

- Introduction to behavioral security: the human factor in cybersecurity
- Social engineering: phishing, vishing, spear-phishing, pretexting
- Psychological principles in security: cognitive biases, persuasion
- Insider threats: malicious vs. negligent insiders
- Behavioral analytics and User Entity and Behavior Analytics (UEBA)
- Security culture and awareness training design
- Organizational policies and procedures to mitigate behavioral risks
- Ethical hacking: social engineering penetration testing
- Case studies: notable social engineering and insider threat incidents

Learning Outcomes

- Analyze human factors that contribute to security breaches
- Identify and counter social engineering attacks
- Design security awareness training programs
- Implement behavioral analytics tools to detect insider threats

References & Materials

- Hadnagy, C. — Social Engineering: The Science of Human Hacking (2nd ed.)
- Mitnick, K. — The Art of Deception
- Lecture notes

Tools: GoPhish (phishing simulation), Splunk (UEBA), SET (Social Engineering Toolkit)

X-9) CEC448 — System Administration

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Tatsachou Ngningaye D.
Prerequisites: CEC220 Principles of Operating Systems, CEC209 Introduction to Networking and Security

Course Description

This course covers the skills and knowledge required to administer Linux and Windows server systems in an enterprise environment. Students will learn installation, configuration, user management, monitoring, backup, and automation of system administration tasks.

Objectives

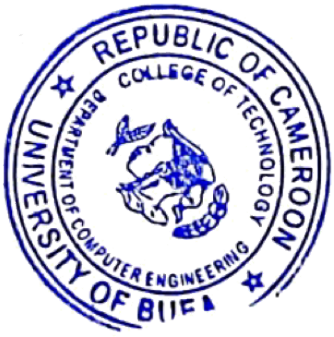
- Install, configure, and manage Linux and Windows servers
- Manage users, groups, permissions, and system resources
- Configure network services: DNS, DHCP, FTP, SSH, web servers
- Automate administrative tasks using scripts
- Monitor system performance and ensure high availability

Course Content

- Linux administration: filesystem, users, groups, permissions
- Package management: apt, yum, dnf
- Shell scripting for automation: Bash scripts, cron jobs
- Network services: DNS (BIND), DHCP, NFS, Samba, Apache/Nginx
- SecurityContainer orchestration: Kubernetes architecture, pods, deployments, services, Helm
- Windows Server: Active Directory, Group Policy, WSUS
- Virtualization: VMware ESXi, KVM, Hyper-V
- Backup and recovery strategies
- System monitoring: Nagios, Zabbix, Grafana
- Containerization basics: Docker administration

Learning Outcomes

- Install and configure Linux and Windows server systems
- Manage network services and user authentication
- Write Bash scripts to automate administrative tasks



- Monitor system health and implement backup strategies

References & Materials

- Evi Nemeth et al. — UNIX and Linux System Administration Handbook (5th ed.)
- Red Hat Enterprise Linux documentation
- Lecture notes

Tools: Ubuntu Server, CentOS, Windows Server, VMware, Docker, Nagios, Zabbix

X-10) CEC449 — Cybersecurity Principles

Semester: 1 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Tatsachou Ngningaye D.
Prerequisites: CEC461 Information Systems Security, CEC304 Data Security and Integrity

Course Description

This course provides a comprehensive foundation in cybersecurity principles, frameworks, and practices. Students will study offensive and defensive security techniques, penetration testing methodology, vulnerability management, and incident response in depth.

Objectives

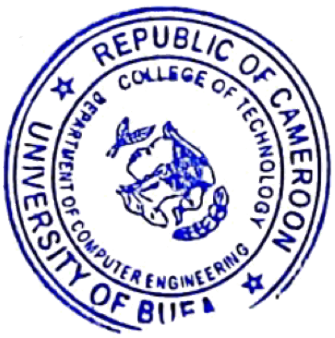
- Understand cybersecurity frameworks and standards
- Study offensive security: reconnaissance, exploitation, post-exploitation
- Apply defensive security techniques: hardening, monitoring, detection
- Conduct structured vulnerability assessments and penetration tests

Course Content

- Cybersecurity fundamentals: threats, vulnerabilities, attacks
- NIST, CIS Controls, and ISO 27001 frameworks
- Offensive security: OSINT, network scanning, vulnerability exploitation
- Web application security: OWASP Top 10
- Penetration testing methodology: planning, reconnaissance, exploitation, reporting
- Defensive security: intrusion detection/prevention systems (IDS/IPS)
- Security information and event management (SIEM)
- Digital forensics: evidence collection, chain of custody
- Malware analysis: static and dynamic techniques
- Incident response planning and execution

Learning Outcomes

- Conduct structured vulnerability assessments
- Perform basic penetration tests following a documented methodology
- Deploy and configure IDS/IPS and SIEM tools
- Execute an incident response process
- Analyze malware samples in a safe environment



References & Materials

- Engebretson, P. — The Basics of Hacking and Penetration Testing (2nd ed.)
- Weidman, G. — Penetration Testing (No Starch Press)

Tools: Kali Linux, Metasploit, Nmap, Burp Suite, Wireshark, Splunk, Autopsy

X-11) CEC450 — Enterprise Core Networking

Semester: 2 **Credits:** 4 **Department:** Computer Engineering **Course Master:** Kome Kole
Prerequisites: Introduction to Networks, Switching & Routing Essentials, Python for Networking, REST APIs basics

Course Description

This course provides an in-depth exploration of network automation using Ansible, Terraform, and REST API authentication techniques. Students will implement Infrastructure as Code (IaC) principles and network orchestration, gaining hands-on experience through Cisco DevNet labs.

Objectives

- Understand Ansible and Terraform for network automation

- Learn Infrastructure as Code (IaC) principles
- Implement network automation using Ansible Playbooks
- Use Terraform for Cisco IOS XE and RESTCONF automation
- Understand REST API authentication: Basic Auth, API Keys, OAuth, Access Tokens

Course Content

- Introduction to Network Automation and Infrastructure as Code
- Ansible for Network Automation: architecture, Playbooks, Modules, Inventories
- Hands-on: Automating Cisco Device Configurations with Ansible
- Terraform for Network Automation: HCL, architecture, workflow
- Hands-on: Automating Cisco IOS XE with Terraform and RESTCONF
- REST API Authentication: Basic Auth, API Keys, OAuth, Access Tokens
- Hands-on: Configuring REST API Authentication in Network Devices
- Case Study: Automating Cisco Network Infrastructure
- Final Project: Deploying Configurations with Ansible & Terraform

Learning Outcomes

- Implement Infrastructure as Code to manage network devices
- Develop Ansible Playbooks for network configuration management
- Automate Cisco network operations using Terraform
- Configure REST API authentication mechanisms
- Troubleshoot network automation issues

References & Materials

- Geerling, J. — Ansible for DevOps
- Brikman, Y. — Terraform: Up & Running
- Edelman et al. — Network Programmability and Automation
- Cisco DevNet Documentation

Tools: Ansible, Terraform, Cisco Packet Tracer, GNS3, Cisco DevNet Sandbox, Postman, Wireshark

XI. COURSE DESCRIPTIONS — LEVEL 600 (M.TECH)

XI-1) CEC601 — Advanced Software Development

Semester: 1 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Kometa Denis Mbowoh
Prerequisites: Data Structures & Algorithms, Systems Analysis and Design, Programming Languages, Database Principles, UML, Software Architecture

Course Description

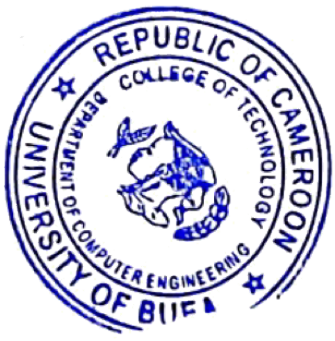
This course covers advanced software development principles, methods, and tools. Students will design, develop, and evaluate a commercially realistic software system, learning advanced practices including DevOps, CI/CD, and collaborative team management.

Objectives

- Master advanced software development principles and methods
- Design, develop, and evaluate software meeting functional and non-functional requirements
- Learn advanced practices through hands-on projects
- Apply principles of good software design including data abstraction

Course Content

- Overview of Software Engineering
- Introduction to Software Engineering Project Management
- Team Formation and Management
- Software Development Environments and Tools
- Requirements Engineering and Management
- Software Design: architectural and detailed design
- Software Construction: coding standards, refactoring, code review
- Software Evaluation: testing strategies and quality assurance
- Software Deployment: CI/CD, DevOps practices



- Tutorials

Learning Outcomes

- Elicit and manage customer requirements throughout the project
- Design, construct, and evaluate software to meet customer requirements
- Plan and manage software development to completion
- Collaborate with team members and monitor team health
- Communicate project, software, and team information to all stakeholders

References & Materials

- Smart, J.F. — BDD in Action: Behaviour-Driven Development (Manning, 2015)
- Bass, L. et al. — DevOps: A Software Architect's Perspective (Addison-Wesley)
- Bryant & Marin-Perez — Continuous Delivery in Java (O'Reilly)

Tools: DevOps Tools Kit, StarUML, VS Code, PyCharm, NetBeans, Eclipse, Docker, AWS

XI-2) CEC603 — Information Systems and Network Security

Semester: 1 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Prof. Sone M. (part-time)
Prerequisites: CEC461 Information Systems Security, networking fundamentals, cryptography basics

Course Description

This advanced course covers the comprehensive management of information and network security in enterprise environments. Students will study advanced threat modeling, security architecture, penetration testing, security operations, and governance frameworks at the postgraduate level.

Objectives

- Apply advanced threat modeling and risk assessment techniques
- Design enterprise security architectures
- Conduct advanced penetration testing and security assessments
- Manage security operations centers (SOC) functions
- Apply security governance frameworks: ISO 27001, NIST, COBIT

Course Content

- Advanced Threat Modeling: STRIDE, PASTA, LINDDUN
- Enterprise Security Architecture: SABSA, TOGAF security
- Advanced Cryptography: elliptic curve cryptography, post-quantum
- Security Operations Center (SOC): tools, processes, metrics
- Advanced Network Security: micro-segmentation, zero-trust
- Cloud Security Architecture: CASB, CWPP, CSPM
- Advanced Penetration Testing: red team operations
- Digital Forensics and Incident Response (DFIR) at scale
- Security Governance: ISO 27001, NIST CSF, compliance
- Case studies: major security breaches and lessons learned

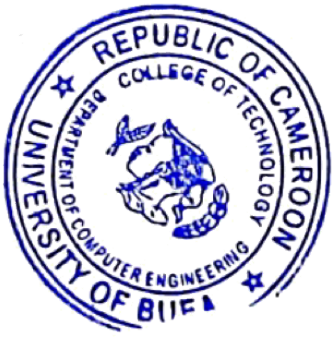
Learning Outcomes

- Conduct advanced threat modeling and risk assessments
- Design and evaluate enterprise security architectures
- Lead security operations and incident response
- Apply and audit security governance frameworks

References & Materials

- Stallings — Cryptography and Network Security (7th ed.)
- NIST SP 800 series
- ISO/IEC 27001:2022 standard

Tools: Kali Linux, Metasploit, Splunk Enterprise, Nessus Professional, Wireshark, Zeek



XI-3) CEC605 — Cloud Architecture

Semester: 1 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Kometa Denis Mbowoh
Prerequisites: Systems Analysis and Design, Programming Languages, Database Management, Computer Networking, Computer Security

Course Description

This advanced course provides an in-depth understanding of cloud computing principles, architectures, and design decisions. Students will gain skills to become Cloud Architects, capable of proposing and validating advanced cloud architectural solutions.

Objectives

- Gain an advanced overview of cloud computing principles and architecture
- Understand design decisions for cloud service models (SaaS, PaaS, IaaS)
- Choose suitable cloud service providers and services for given projects
- Build and deploy applications in a cloud environment
- Explain privacy and security risks in cloud computing

Course Content

- Cloud Computing Foundation: history, concepts, evolution
- Technologies that led to Cloud Computing: virtualization, containers
- Types of Clouds and Cloud Architectures
- Using Cloud Platforms: AWS, Azure, GCP
- Exploring Cloud Infrastructures: compute, storage, networking
- Cloud Technologies: Web services, APIs, microservices
- Cloud Services and Applications: serverless, containers
- Cloud Security: shared responsibility, IAM, encryption
- Case Study on Open Source and Commercial Clouds
- Mini Class Projects and Tutorials

Learning Outcomes

- Build a solid foundation in advanced cloud computing concepts
- Gain skills to become a Cloud Architect
- Propose and validate systematic Cloud Architectural Solutions
- Obtain hands-on experience with orchestration tools
- Design, build, and deploy real-world applications in the Cloud

References & Materials

- Erl, T. — Cloud Computing: Concepts, Technology & Architecture (Prentice Hall, 2013)
- Kavis, M. — Architecting the Cloud (Wiley, 2014)
- Rajkumar et al. — Cloud Computing Principles and Paradigms (Wiley, 2011)

Tools: VMware, VirtualBox, AWS, Microsoft Azure, Google Cloud, Docker, Kubernetes

XI-4) CEC621 — Artificial Intelligence

Semester: 1 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Dr. Tchapga Tchito Christian
Prerequisites: CEC323 Introduction to AI, Machine Learning, Statistics/Probability

Course Description

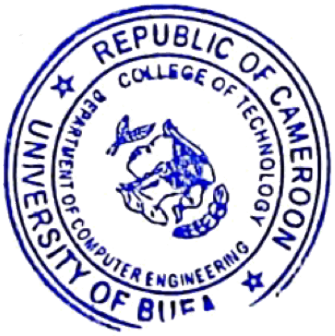
This advanced postgraduate course covers cutting-edge AI techniques and their applications. Students will study advanced machine learning, natural language processing, computer vision, planning, and multi-agent systems, with a focus on implementing and evaluating AI systems at scale.

Objectives

- Study advanced AI algorithms and their theoretical foundations
- Implement and evaluate NLP and computer vision systems
- Understand multi-agent systems and AI planning
- Critically evaluate the ethical and societal impact of AI

Course Content

- Advanced Search and Optimization: A*, simulated annealing, genetic algorithms



- Probabilistic Reasoning: Bayesian networks, Markov models, HMMs
- Machine Learning: advanced supervised learning, ensemble methods
- Neural Networks: advanced architectures — attention, transformers
- Natural Language Processing: text classification, NER, machine translation, LLMs
- Computer Vision: image classification, object detection, segmentation
- Planning: STRIPS, PDDL, and HTN planning
- Multi-agent Systems: game theory, negotiation, cooperation
- AI Ethics: fairness, accountability, transparency, bias
- Research seminar: current trends in AI

Learning Outcomes

- Implement advanced AI algorithms and evaluate their performance
- Build NLP and computer vision applications
- Understand probabilistic AI and its applications
- Critically evaluate AI systems from ethical perspectives

References & Materials

- Russell & Norvig — Artificial Intelligence: A Modern Approach (4th ed.)
- Goodfellow et al. — Deep Learning (MIT Press)
- Course research papers

Tools: Python, TensorFlow, PyTorch, Hugging Face, OpenCV, NLTK/SpaCy, Jupyter

XI-5) CEC610 — Utility Computing

Semester: 2 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Kometa Denis Mbowoh
Prerequisites: Computer Systems, Networking, Cloud Computing, Software Design, Database Principles

Course Description

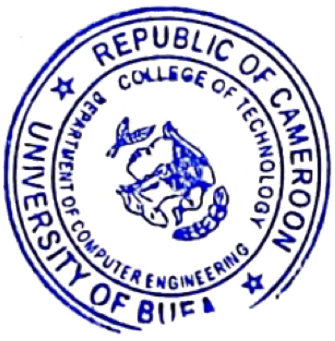
This course covers utility computing services, models, and infrastructure. Students will learn the technology behind utility computing, transition planning from traditional IT to utility models, and the business and cultural implications of utility computing adoption.

Objectives

- Understand and master utility computing services, models, and infrastructure
- Learn the technology and concepts behind utility computing
- Understand step-by-step transition plans for major IT services
- Understand the business implications of utility computing strategy

Course Content

- Overview of Utility Computing
- Service Level Agreements (SLA): design and management
- The Utility Computing Reference Model
- Overview of a Transformational Approach to IT
- Utility Computing Technologies
- Software Application Services Framework
- Business implications: cultural change, adoption strategy
- Case Studies and Tutorials



Learning Outcomes

- Master utility computing services, models, SLA, and infrastructure
- Understand the technologies and concepts behind utility computing
- Develop transition plans for major IT services
- Address business and cultural implications of utility computing adoption

References & Materials

- Mendoza, A. — Utility Computing Technologies, Standards, and Strategies (Artech House, 2007)
- Bunker, G. & Thomson, D. — Delivering Utility: Computing Business-driven IT Optimization (Wiley, 2016)

Tools: VMware, VirtualBox, AWS, Microsoft Azure, Google Cloud, IDEs

XI-6) CEC612 — FPGA Based Embedded Systems

Semester: 2 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Prof. Moffo Lonla Bertrand
Prerequisites: CEC217 Computer Architecture, EEC330 Basics of Digital Electronics

Course Description

This course covers the design and implementation of embedded systems using Field Programmable Gate Arrays (FPGAs). Students will learn hardware description languages (HDL), FPGA architecture, and design flows to implement digital systems, signal processing algorithms, and system-on-chip designs.

Objectives

- Understand FPGA architecture and its role in embedded systems
- Design digital systems using VHDL or Verilog
- Implement and verify FPGA designs using industry EDA tools
- Design complete SoC systems using FPGAs

Course Content

- Introduction to FPGAs: architecture, CLBs, DSP blocks, memory
- VHDL fundamentals: entities, architectures, processes, signals
- Combinational logic design in VHDL/Verilog
- Sequential logic: FSMs, registers, counters
- FPGA design flow: synthesis, place and route, timing analysis
- IP cores and reusable design components
- FPGA-based signal processing: FIR filters, FFT implementation
- Memory interfaces: BRAM, DDR, flash
- High-Level Synthesis (HLS): C to hardware
- SoC design with MicroBlaze or ARM-based soft processors
- Project: complete FPGA-based embedded system

Learning Outcomes

- Design and simulate digital systems using VHDL or Verilog
- Implement designs on Xilinx or Intel (Altera) FPGAs
- Apply FPGAs for signal processing applications
- Design SoC systems with embedded processor cores

References & Materials

- Pong, P.C. — FPGA Prototyping by VHDL Examples (Xilinx, 2008)
- Maxfield, C. — The Design Warrior's Guide to FPGAs
- Lecture notes

Tools: Xilinx Vivado, Intel Quartus, ModelSim, MATLAB HDL Coder

XI-7) CEC616 — Deep Learning

Semester: 2 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Dr. Tchapga Tchito Christian
Prerequisites: CEC424 or CEC621 (AI), Machine Learning, Python, Mathematics (linear algebra, calculus)

Course Description

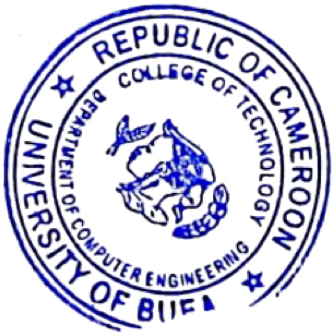
This advanced course covers state-of-the-art deep learning architectures and training techniques. Students will implement and evaluate CNNs, RNNs, Transformers, and GANs, and apply them to real-world problems in vision, language, and other domains.

Objectives

- Master advanced deep learning architectures
- Implement and train deep neural networks on large datasets
- Apply regularization, optimization, and transfer learning techniques
- Evaluate and interpret deep learning models

Course Content

- Review of neural networks and backpropagation
- Convolutional Neural Networks (CNNs): architectures (ResNet, VGG, EfficientNet)
- Transfer learning and fine-tuning pre-trained models



- Recurrent Neural Networks (RNNs): LSTM, GRU
- Sequence-to-sequence models and attention mechanisms
- Transformer architectures: BERT, GPT overview
- Generative models: VAEs, GANs, diffusion models
- Optimization: Adam, learning rate schedules, batch normalization
- Regularization: dropout, weight decay, data augmentation
- Model interpretability: SHAP, LIME, Grad-CAM
- Project: applying deep learning to a real-world dataset

Learning Outcomes

- Implement advanced CNN and RNN architectures
- Apply transfer learning to domain-specific problems
- Train and evaluate GANs and generative models
- Interpret and explain deep learning model predictions

References & Materials

- Goodfellow et al. — Deep Learning (MIT Press, 2016)
- Chollet — Deep Learning with Python (2nd ed.)
- Papers from NeurIPS, ICML, ICLR

Tools: Python, TensorFlow 2.x, Keras, PyTorch, Google Colab, Weights & Biases

XI-8) CEC618 — Advanced Data Structures and Algorithms

Semester: 2 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Dr. Nyanga Bernard
Prerequisites: CEC305 Data Structure and Algorithm, Mathematics, Programming proficiency

Course Description

This advanced course covers sophisticated data structures and algorithm design paradigms needed for high-performance computing and complex problem solving. Students will study amortized analysis, advanced graph algorithms, string algorithms, and computational geometry.

Objectives

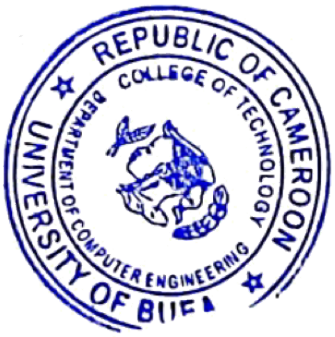
- Master amortized analysis and advanced complexity theory
- Study and implement advanced graph and string algorithms
- Understand probabilistic and approximation algorithms
- Apply advanced algorithmic techniques to competitive programming problems

Course Content

- Amortized analysis: aggregate, accounting, potential methods
- Advanced trees: red-black trees, splay trees, B-trees, segment trees
- Advanced heaps: Fibonacci heaps, binomial heaps
- Advanced graph algorithms: max flow (Ford-Fulkerson), min-cut
- String algorithms: KMP, Rabin-Karp, suffix arrays and trees
- Computational geometry: convex hull, sweep line
- Randomized algorithms: hashing, randomized quicksort, skip lists
- Approximation algorithms: greedy, LP relaxation
- Parametric search and fractional cascading
- Competitive programming: problem-solving strategies

Learning Outcomes

- Analyze algorithm complexity using amortized analysis
- Implement advanced tree and heap structures
- Apply max-flow algorithms to network optimization problems
- Solve competitive programming challenges using advanced techniques



References & Materials

- Cormen et al. — Introduction to Algorithms (3rd ed.)
- Dasgupta et al. — Algorithms (Melvil Press)
- Competitive programming resources (Codeforces, LeetCode)

Tools: C++, Python, competitive programming judges (Codeforces, AtCoder), VS Code

XI-9) CEC691 — Research Methodology and Scientific Writing

Semester: 1 (Year 5) **Credits:** 6 **Department:** Computer Engineering **Course Master:** Dr. Nyanga Bernard
Prerequisites: Enrollment in M.Tech programme

Course Description

This course develops the research and scientific communication skills necessary for postgraduate students. Students will learn to identify research problems, conduct literature reviews, design research methodologies, analyze data, and write and present research findings in academic formats.

Objectives

- Identify and formulate research problems
- Conduct systematic literature reviews
- Design appropriate research methodologies
- Write research papers following academic conventions
- Present research findings professionally

Course Content

- Introduction to research: types of research, research process
- Problem formulation and research questions/hypotheses
- Literature review: search strategies, critical evaluation, synthesis
- Research design: quantitative, qualitative, mixed methods
- Data collection methods: surveys, experiments, case studies
- Data analysis: statistical methods, qualitative analysis
- Research ethics: plagiarism, data integrity, IRB
- Scientific writing: structure of a thesis, journal article, conference paper
- Academic citation: IEEE, APA, Harvard referencing styles
- Oral presentation and defense skills

Learning Outcomes

- Formulate a researchable problem with clear questions and hypotheses
- Conduct a systematic literature review
- Design and execute an appropriate research methodology
- Write a research paper or thesis chapter in proper academic format
- Present research findings convincingly

References & Materials

- Creswell, J. — Research Design: Qualitative, Quantitative & Mixed Methods (5th ed.)
- UB Thesis Guide
- IEEE Author Tools

Tools: Mendeley/Zotero (reference management), LaTeX, Microsoft Word, SPSS, Python/R

XI-10) COT602 — Big Data Analytics

Semester: 2 **Credits:** 6 **Department:** Computer Engineering **Course Master:** Dr. Mih T. (part-time)
Prerequisites: Data Structures and Algorithms, Problem-Solving, Business thinking

Course Description

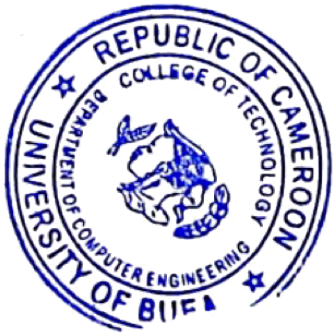
This course enables students to understand and apply big data analytics frameworks. Students will learn big data concepts, MapReduce, Hadoop/Spark, and apply these tools to real datasets from telecommunications, power systems, and electronics.

Objectives

- Understand the framework for analyzing big data
- Understand the principles of big data, data science, and data analytics
- Expose students to real data analysis systems in telecom, power, and electronics

Course Content

- Chapter 1: Introduction to Big Data Analytics — concepts, MapReduce, NoSQL
- Chapter 2: Practical Tutorial on MapReduce and Hadoop
- Chapter 3: Developing a Strategy for Integrating Big Data Analytics into the Enterprise



- Chapter 4: Big Data Analytics Tools — HDFS, MapReduce, Hive, Pig, HBase, Mahout
- Chapter 5: Developing Big Data Analytics Applications
- Chapter 6: Hadoop and Spark Projects in Telecom, Electronics and Power Systems
- Chapter 7: Mathematical Modeling of Big Data Analysis

Learning Outcomes

- Gain a strong foundation in big data, data science, and MapReduce
- Develop skills in Hadoop, MapReduce, and Spark
- Integrate big data analytics into enterprise scenarios
- Apply big data techniques to real-world domain problems

References & Materials

- Zikopoulos & Eaton — Understanding Big Data: Analytics for Enterprise Class Hadoop
- Kitchin, R. — The Data Revolution (SAGE Publications)
- Prajapati — Big Data Analytics with R and Hadoop (Packt, 2015)

Tools: Cloudera, PyCharm, Anaconda, Jupyter Notebook, RStudio, Apache Hadoop, Apache Spark

XI-11) CEC698 — M.Tech Dissertation

Semester: 2 (Year 5) **Credits:** 42 **Department:** Computer Engineering **Course Master:** All Staff
Prerequisites: Successful completion of all taught modules; CEC696 Pre-defense Seminar

Course Description

The M.Tech Dissertation is the culminating research activity of the Software Engineering postgraduate programme. Students will independently conduct original research on a topic related to Computer Engineering, produce a formal dissertation, and defend their work before an academic committee.

Objectives

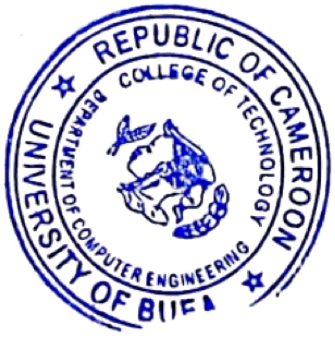
- Conduct independent, original research in Computer Engineering
- Produce a substantial written dissertation meeting academic standards
- Demonstrate command of research methodology and the chosen research domain
- Successfully defend the dissertation before an examination committee

Course Content

- Research proposal development and approval
- Literature review and gap analysis
- Research design and methodology implementation
- Data collection, analysis, and interpretation
- Dissertation writing: introduction, literature review, methodology, results, discussion, conclusion
- Internal review and feedback iterations
- Pre-defense seminar (CEC696) as a prerequisite
- Final dissertation submission
- Oral defense before examination committee

Learning Outcomes

- Produce a dissertation on a researchable topic in Computer Engineering
- Demonstrate ability to conduct rigorous, independent research
- Present and defend research findings before an academic committee
- Contribute original knowledge to the field of Computer Engineering



References & Materials

- UB Student's Thesis Guide
- IEEE formatting guidelines
- Field-specific references as directed by supervisor

Tools: Project-specific tools as directed by research supervisor; LaTeX or Word for writing

XII. COURSES OFFERED TO OTHER DEPARTMENTS

XII-1) CEC214 — Object Oriented Programming Languages

Semester: TBD **Credits:** TBD **Department:** Computer Engineering **Course Master:** Dr. Melingui Melono Robert

Prerequisites: Basic programming knowledge

Course Description

This course introduces object-oriented programming concepts for students from other departments. Students will learn to apply OOP principles (encapsulation, inheritance, polymorphism) using a high-level programming language to solve problems relevant to their discipline.

Objectives

- Understand object-oriented programming fundamentals
- Apply OOP concepts to solve domain-specific problems
- Develop software using classes, objects, and inheritance

Course Content

- Introduction to OOP: classes, objects, methods
- Encapsulation, inheritance, and polymorphism
- Exception handling and file I/O
- Collections and basic data structures
- Domain-specific application development

Learning Outcomes

- Design and implement classes for domain-specific applications
- Apply OOP principles in software development
- Debug and test object-oriented programs

References & Materials

- Department-specific lecture notes

Tools: Java or Python, Eclipse/PyCharm, VS Code

XII-2) CEC328 — Microprocessor/Microcontroller in Mechatronics

Semester: TBD **Credits:** TBD **Department:** Computer Engineering **Course Master:** Tchinda Tandjon Valerry

Prerequisites: Basic electronics and programming knowledge

Course Description

This course introduces microprocessors and microcontrollers in the context of Mechatronics systems. Students will learn to program microcontrollers, interface with sensors and actuators, and implement control algorithms for electro-mechanical systems.

Objectives

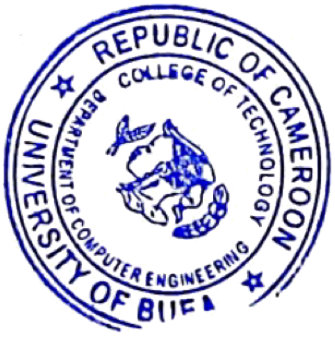
- Understand microprocessor and microcontroller architecture
- Program microcontrollers for mechatronics applications
- Interface microcontrollers with sensors, actuators, and mechatronic components
- Implement control algorithms in embedded C

Course Content

- Microcontroller architecture and programming
- Digital and analog I/O for mechatronic systems
- Motor control: DC, servo, stepper motors
- Sensor interfacing: encoders, IMUs, proximity sensors
- PID control implementation on microcontrollers
- Communication protocols: UART, SPI, I2C, CAN
- Real-time control for mechatronics

Learning Outcomes

- Program a microcontroller for a mechatronics application
- Interface sensors and actuators with a microcontroller



- Implement a PID controller on embedded hardware

References & Materials

- Mazidi et al. — AVR Microcontroller and Embedded Systems
- Lecture notes

Tools: Arduino IDE, MATLAB/Simulink, Proteus, C compiler

XII-3) CEC405 — Microprocessor/Microcontroller Practicals

Semester: TBD **Credits:** TBD **Department:** Computer Engineering **Course Master:** Prof. Moffo Lonla Bertrand

Prerequisites: CEC328 or equivalent theory course

Course Description

This practical course provides hands-on experience with microprocessor and microcontroller programming and interfacing. Students will implement a series of lab exercises culminating in a capstone embedded systems project.

Objectives

- Develop practical skills in microcontroller programming
- Interface a variety of peripheral devices with microcontrollers
- Debug and test embedded systems through hands-on laboratory work

Course Content

- Lab 1: GPIO — LED control and button interfacing
- Lab 2: Timer and PWM — servo motor control
- Lab 3: ADC — analog sensor reading
- Lab 4: UART communication
- Lab 5: I2C and SPI protocols
- Lab 6: Interrupt handling
- Lab 7: Motor driver interfacing
- Lab 8: Capstone project — embedded system design and implementation

Learning Outcomes

- Complete a series of embedded systems lab exercises
- Interface diverse peripherals with a microcontroller
- Design and implement a capstone embedded project

References & Materials

- Lab handouts and Arduino documentation

Tools: Arduino, Raspberry Pi, multimeter, oscilloscope, breadboard, electronic components

