Course Outline

ICS 2309 - Commercial Programming & BCT 2315 - Computer Systems Project
Lecturer: Dr. Lawrence Nderu
Course Type: Practical, Industry-Oriented

Course Overview
This combined course integrates Commercial Programming (ICS 2309) and Computer Systems Project (BCT 2315), providing students with real-world software development experience. It follows modern software engineering principles, equipping students with skills in agile methodologies, DevOps, cloud deployment, security, testing, and documentation.

Students will develop a complete software product, ensure industry-level documentation, and publish a research paper based on their project. Additionally, they will earn industry certifications, engage in peer code reviews, attend tech talks and industry visits, and submit progress updates.

Course Learning Objectives
By the end of the semester, students will:

1. Design and develop scalable, secure, and maintainable software applications.
2. Apply modern software development and testing practices aligned with industry standards.
3. Implement CI/CD pipelines, DevOps methodologies, and cloud deployments.
4. Develop a fully functional, industry-grade software product with proper documentation.
5. Write and publish a research paper based on the work in this course
6. Gain exposure to real-world software development through industry visits, guest lectures, and certifications.
7. Improve collaboration, project management, and professional communication skills.

Course Delivery & Assessment
This course follows a highly practical, project-based learning approach.

Evaluation will include:

- Software Product & Documentation
- Research Paper Submission
- Sit-In Examination
- Weekly Progress Reports & Peer Code Reviews
- Industry Engagement & Continuous Learning (includes blogs on tech talks, certifications, and industry visits)
- Final Presentations & Industry Evaluation

Breakdown

Week 1: Introduction to Modern Software Development & Project Ideation

- Overview of Modern Software Development Paradigms (Agile, DevOps, Lean Software Development).
- Understanding 21st Century Software Development best practices.
- Introduction to Commercial Programming & Enterprise Software.

- Project ideation: Defining a problem statement and selecting a domain.
- Forming teams and assigning roles (Product Owner, Lead Developer, UI/UX Designer, Backend Engineer, etc.).
- Introduction to Project Management Tools (Jira, Trello, GitHub Projects).

Deliverables:

✔ Project Proposal

✔ Team formation & role assignment

Reference:

- Sommerville, I. (2020). *Software Engineering (10th ed.).* Pearson.

Week 2: Software Design & Architecture

- Software Design Patterns: MVC, MVVM, Microservices.
- Scalability & Performance Considerations in Software Design.
- Security in Software Design: Secure authentication, role-based access control, and encryption.
- Wireframing & Prototyping using Figma or Adobe XD.
- Database Design & Optimization (SQL vs. NoSQL).

Deliverables:

✔ Software Architecture Diagram

✔ Wireframes & Initial UI Design

References:

- Gamma, E. et al. (2023 --). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

Week 3: Development Environments & DevOps

- Version Control & Collaboration (Git, GitHub, GitLab, Bitbucket).
- Continuous Integration/Continuous Deployment (CI/CD) Pipelines (GitHub Actions, Jenkins, Docker, Kubernetes).
- Choosing a Tech Stack (MERN, Django, .NET, etc.).

Deliverables:

✔ GitHub Repository Setup

✔ CI/CD Pipeline Configured

References:

- Chacon, S., & Straub, B. (2021). *Pro Git (2nd ed.).* Apress.

Week 4: Frontend & Backend Development

- Frontend Development: React.js, Vue.js, Angular.
- Backend Development: Node.js, Django, Flask, Spring Boot.
- API Development & Documentation (Swagger, Postman).

Deliverables:

✔ Working Frontend & Backend Prototype

✔ API Documentation

Week 5-6: Software Testing & Debugging

- Unit Testing & Integration Testing (JUnit, Mocha, Jest, PyTest).

- End-to-End Testing (Selenium, Cypress).
- Debugging Techniques & Code Review Best Practices.

Deliverables:
✔ Unit & Integration Tests for core functionality

Week 7: Security & Performance Optimization
- Secure Coding Practices (Preventing SQL Injection, XSS, CSRF).
- Performance Optimization (Profiling, Caching, Load Balancing).

Deliverables:
✔ Security Audit Report

Week 8-9: Deployment & DevOps in Practice
- Deploying Applications (AWS, Heroku, Firebase, DigitalOcean).
- Infrastructure as Code (Terraform & Ansible).
- Monitoring & Logging (Grafana, Prometheus, ELK Stack).

Deliverables:
✔ Live Demo of the Software Running on the Cloud

Week 10-11: Final Development Sprint & System Integration
- Refactoring & Code Optimization for maintainability.
- Final UI/UX Enhancements.
- Database Optimization & Indexing Strategies.

Deliverables:
✔ Finalized Software in near-production state

Week 12: Research Paper Writing & Documentation
- Technical Writing & Academic Paper Structuring.
- Research Methodologies in Software Engineering.

Deliverables:
✔ Research Paper – This should be done thought out the semester

References:

- Day, R. A. (2023 --). *How to Write and Publish a Scientific Paper (7th ed.).* Cambridge University Press.

Week 13: Final Presentations, Peer Reviews & Testing
- Final software demonstrations.
- Peer Code Reviews & Testing.
- Research Paper Finalization.

Deliverables:
✔ Research Paper submitted

Week 14: Final Evaluation & Industry Feedback
- Industry experts review software & research paper.
- Students present their final work.

- Submission of all deliverables.

Final Deliverables:

✓ Fully Functional Software Product
✓ Industry-Level Documentation
✓ Published Research Paper

Philosophy of the Class

This course is designed to be a highly practical, industry-aligned learning experience, ensuring that students not only gain theoretical knowledge but also acquire real-world, hands-on expertise in modern software development. The course is structured to mirror industry best practices, enabling students to build scalable, secure, and maintainable software solutions while fostering a research-driven mindset.

By the end of the semester, students will:

- Develop real-world, deployable software that meets industry standards.
- Learn modern software engineering, testing, and DevOps skills, preparing them for the evolving tech landscape.
- Publish a research paper based on their project, contributing to the academic and professional knowledge base.
- Gain industry-recognized certifications, attend tech talks, and write technical blogs, ensuring continuous professional growth.
- Become industry-ready with complete documentation and cloud deployment experience, making them valuable assets to any tech-driven organization.

This philosophy emphasizes a learning-by-doing approach, bridging the gap between academic knowledge and industry expectations, equipping students with future-proof skills in software engineering and innovation.