NATIONAL UNIVERSITY OF ENGINEERING
COLLEGE OF SCIENCES

COMPUTER SCIENCE PROGRAM

CC344 – SOFTWARE DEVELOPMENT

I. GENERAL INFORMATION

CODE	: CC344 Software Development
SEMESTER	: 6
CREDITS	: 4
HOURS PER WEEK	: 6 (Theory, Practice)
PREREQUISITES	: CC341 Software Engineering
CONDITION	: Mandatory

II. COURSE DESCRIPTION

The course prepares students for the development and implementation of software applications for diverse purposes. Students understand that software development is the process of conceiving, specifying, designing, programming, documenting, testing, and bug fixing involved in creating and maintaining applications, frameworks, or other software components pointing to satisfy and fulfill requirements and needs of specific client/business or perceived needs of some set of potential users. Students understand the different approaches, methodologies, processes and models for software development throughout its lifecycle. Along the course, students design, develop and implement a software application whose functionality is presented and defended.

III. COURSE OUTCOMES

At the end of the course students:

1. Identify needs and requirements to be solved by software applications.
2. Understand and apply the stages of software development for designing reliable, secure, resilient and high-performance software applications.
3. Define software architectures appropriate to the particular characteristics of the problem and the proposed solution.
4. Design and implement software applications integrating web services, database access and friendly user interfaces.

IV. LEARNING UNITS

1. INTRODUCTION

Introduction and basic concepts. Software lifecycle. Software development process. Software engineering. Standards and best practices for software development. Web applications development.

2. STAGES OF SOFTWARE DEVELOPMENT

Problem analysis. Market research. Requirements gathering for the proposed business solution. Devising a plan or design for the software-based solution. Software implementation (coding). Software testing. Deployment. Documentation. Maintenance and bug fixing

3. SOFTWARE DESIGN

Design concepts: abstraction, refinement, modularity, software architecture, control hierarchy, structural partitioning, data structure, software procedure, information hiding. **Design considerations:** compatibility, extensibility, modularity, fault-tolerance, maintainability, reliability, reusability, robustness, security, usability, performance, portability, scalability. Modeling language: UML, structural and behavioral software description. **Design patterns:** Domain-specific patters, creational patterns, structural patterns, behavioral patterns, concurrency patterns. **User-interface design**.

4. SERVER-SIDE PROGRAMMING

Distributed applications. Client-server structure and functioning. Client host and server host. Syntax. Variable and operators. Web applications. Object oriented programming. Application programming.

5. DATA BASE ACCESS

Controllers and addresses. Syntax. Data basic access. Transactions. Concurrency. Meta-data. Storage structures. Redundancy. Application program interface.

6. WEB SERVICES

Introduction. HTTP. Markup languages. Web API. W3C. Regression testing web services. Web services frameworks. Web service protocols. Web service specifications. Service oriented architecture. Web map service. XML-RPC. SOAP. WSDL. UDDI.

7. SECURITY IN SOFTWARE DEVELOPMENT

Security aware software development. Security vulnerabilities. Basic principles of software security: protection, requesting, privileges, evidence, configuration. Best practices for software security. Security testing: authentication, authorization, confidentiality, availability, integrity, non-repudiation, and resilience. Web and Internet security.

V. METHODOLOGY

The course takes place in theory, practice and computer laboratory sessions. In theory sessions, the instructor presents the concepts, methodologies, processes and models for software development. In practice session, instructor and students work together for analyzing and proposing solutions to different problems related to software design and development. In computer laboratory sessions, student code and implement software applications using appropriate software development platforms. At the end of the course, each student group present and defend the implemented software application.

VI. EVALUATION FORMULA

The final grade PF is calculated as follows:

$$PF = 0.20\ EP + 0.3\ EF + 0.20\ PR + 0.30\ TF$$

where:

PF: Final grade EP: Mid-term exam EF: Final exam
PR: Practice work: TF: Final project report and defense

VII. BIBLIOGRAPHY

1. The Clean Code: A Code of Conduct for Professional Programmers
   Robert C. Martin
   Prentice Hall Editions, 2018.
2. The Pragmatic Programmer
   David Thomas and Andrew Hunt
   Pearson Editions, 2018