Syllabus for Software Engineering

Description: The focus of this course is to teach students to develop software products considering modern approaches to software engineering: planning, designing, coding, testing, and maintaining and debugging programs. Using the C# programming language and .Net library technologies, students will gain skills in developing desktop applications for the Microsoft Windows operating system and web applications. The main activities that constitute the essence of software engineering are studied and practically applied in the first semester using technologies such as ADO.Net, Entity FW, WPF, and in the second semester, ASP.NET Core MVC.

The objective of the standard discipline "Software Engineering" is to teach students to:

- Apply modern approaches to planning, design, coding, testing, and support and debugging of programs.
- Create and analyze requirements for software products.
- Use UML diagrams to represent program requirements and describe program architecture and design.
- Organize program development using one of the SDLC methodologies.
- Write code in accordance with the main principles of object-oriented design and 'clean code' approaches.
- Develop desktop applications for Windows using WPF technology.
- Develop web applications using ASP.Net technology.
- Develop programs with database access using ADO.Net and Entity FW technology.
- Develop manual and automated tests to ensure code quality.

Topics Part 1:

1. Software Development Life Cycle (SDLC). Team Roles.
   Laboratory Exercise: Team formation, project theme selection, project development methodology choice.
2. Requirements for Software Products. Gathering, Types, Analysis of Requirements.
   Laboratory Exercise: Gathering project requirements, documenting them.
3. UML, Main Types of Diagrams Used in Design and Development.
   Laboratory Exercise: Applying UML diagrams to describe the project and its development.
4. Principles of Object-Oriented Design - SOLID, KISS.
   Laboratory Exercise: Using StyleCop and Code Analysis to support C# coding conventions.
5. Code Analysis with Tools StyleCop and Code Analysis.
   Laboratory Exercise: Describing user interface requirements - mockups, wireframes.
6. ADO.NET Technologies for Data Access. Data Providers.
   Laboratory Exercise: Database design, architectural diagrams.

7. Connected Mode of Database Operation. Connection Strings. Commands, DataReader. Disconnected Mode of Database Operation. DataAdapter.
   Laboratory Exercise: Database generation, access via ADO.Net.
8. Entity Framework, Approaches DBFirst, ModelFirst, Code First. Repository and UnitOfWork Patterns.
   Laboratory Exercise: Creating a three-tier project architecture. Developing a data access layer using Entity Framework.
9. Principles and Tools for User Interface Development, UX.
   Laboratory Exercise: Developing the presentation layer of the project, Graphical Interface using WPF library.
10. WPF Architecture. Hierarchy of Main Classes.
    Laboratory Exercise: Adding necessary graphics, animation to the program.
11. XAML for GUI Development. Event Handling.
    Laboratory Exercise: Developing the business logic layer of the project.
12. WPF Control Elements. Data Binding. WPF Styles, Templates.
    Laboratory Exercise: Covering code with unit tests.
13. Development of WPF Applications Using the MVVM Pattern.
    Test: Applying the MVVM pattern in the WPF part of the program.
14. Inversion of Control, Dependency Injection, Unity.
    Laboratory Exercise: Logging events, remarks, and errors.
15. Software Testing.
    Laboratory Exercise: Presentation and defense of the project.
16. Logger. Logging of Events, Remarks, and Errors.
    Laboratory Exercise: Presentation and defense of the project.

Topics Part 2:

1. Approaches to Designing Complex Systems. Classification of Software Requirements and Their Interrelations.
   Laboratory Exercise: Creating projects. Setting up the environment.
2. Requirements Development Process: Elicitation, Analysis, Specification, Validation.
   Laboratory Exercise: Formulating a list of requirements for Web applications.
3. Software Architecture: Modeling, Architectural Styles.
   Laboratory Exercise: Specification and validation of requirements. Use-case modeling.
4. Architecture of ASP Core MVC Web Applications. Controllers.
   Laboratory Exercise: Delineating subsystems, confirming architecture.
5. Developing Controller Methods.
   Laboratory Exercise: Developing the domain model.
6. Fundamentals of Designing Views.
   Laboratory Exercise: Unit-testing. Implementation of controllers for a specific subsystem.
7. Designing Domain Models. Using Databases.

Laboratory Exercise: Implementation of controllers for a specific subsystem.

8. Designing Typed Views.
   Laboratory Exercise: Implementing methods and their tests for core functionality. Improving Views.
9. Optimization of Controller Methods.
   Laboratory Exercise: Completing the implementation of methods and their tests for core functionality.
10. ASP.NET Core Identity.
    Laboratory Exercise: Refactoring considering Identity.
11. Classification of Software Quality Attributes. Quality Assurance (QA) Techniques.
    Laboratory Exercise: Completing the implementation of the full set of use cases.
12. Quality of Software Development Process.
    Exercise: Improving UI/UX.
13. GoF (Gang of Four) Design Patterns. Structural DP.
    Laboratory Exercise: Enhancing UI/UX.
14. Creational DP.
    Laboratory Exercise: Web application demonstrations by teams and review regarding quality parameters.
15. Behavioral DP.
    Laboratory Exercise: Web application demonstrations by teams and review regarding quality parameters.
16. Refactoring Patterns.
    Laboratory Exercise: Web application demonstrations by teams and review regarding quality parameters.

Signed:

Lesia Klakovych
Associate Professor, Programming Department
Candidate of Physical and Mathematical Sciences
Ivan Franko National University of Lviv
lesya.klakovych@lnu.edu.ua