MS Software Development Syllabus

Program Overview

The MS Software Development program is based on the leading start-up nation high-tech integration program and has been developed exclusively for YU Global.

Students of the intensive program are provided with the unique tools and methodologies needed to research, learn and master new software development technologies and languages.

Students not only acquire high-impact skills through practical software development experience, but also the industry-required soft-skills to foster professionalism through mentoring and teamwork to ensure quality employment opportunities.

Graduates of the program have higher earning potential and are suitable for jobs usually requiring 2-3 years of experience.

Open Lab

All Open Lab courses are required as prerequisites to the specialization tracks.

The Open Lab (OL) encompasses the core fundamentals and pre-requisites that every software developer needs to know, including:

- Essential concepts and behind-the-scenes workings of system programming
- Tools and resources
- Procedural programming languages
- Agile software development methodologies
- Professional skills (effort optimization, accurate time estimation)
- Software development traps & pitfalls
- Soft skills (personal effectiveness, interpersonal skills, communication)
- Technology research skills
- Software development as an individual and as part of a team

Linux Development Environment and Tools 1, Credits:1

This foundational course immerses students in the Linux environment, emphasizing operating system application layers and computer hardware architecture. The curriculum covers essential topics such as memory organization, shell techniques, and process management, providing students with the knowledge necessary to navigate the Linux development landscape.

Foundations of Procedural Programming, Credits: 4

Students will explore the core concepts and principles underpinning procedural programming languages in this introductory course. Key topics include expressions, variables, functions, and file input/output operations. Through hands-on exercises, students will gain a solid understanding of the fundamentals of programming languages.

Implementations of Applied Data Structures 1, Credits: 3

This course delves into the design and implementation of fundamental data structures, with a focus on Abstract Data Types (ADTs) and efficient design practices. Students will engage

with hands-on learning to develop high-performance implementations of various data structures, enhancing their ability to work with complex data.

Fundamentals of System Programming, Credits: 2

In this course, students will explore the intricacies of memory management and memory handling, focusing on topics such as virtual memory and memory segments. Through practical exercises, they will learn to implement memory allocation schemes and avoid common pitfalls.

Linux Development Environment and Tools 2, Credits: 1

Building on the foundation of Linux Development Environment and Tools 1, this course introduces essential tools and paradigms for increasing development process efficiency and flexibility. Students will gain experience with source control, low-level debugging, and testing methodologies.

Algorithm Interfaces and Implementations, Credits: 3

This comprehensive course provides students with an understanding of algorithm fundamentals, focusing on the implementation of common sorting algorithms. Through hands-on exercises, students will learn techniques such as recursion and backtracking, as well as state machine implementation for complex problem-solving.

Implementations of Applied Data Structures 2, Credits: 2

Delving deeper into advanced data structures, this course leverages dedicated algorithms and data structures to enhance students' understanding and proficiency in working with complex data in real-world applications.

Cyber Threat Landscape and System Programming, Credits: 3

This course takes a holistic approach to the cyber threat landscape and advanced system programming. By focusing on cyber-attack methods and techniques used by offensive actors, students will learn the essentials of the Cyber Kill Chain, Exploitation, Visibility & Monitoring. Students will then examine processes, threads, atomics, and synchronization objects, exploring Inter Process Communication (IPC) using signals. They will then implement complex problems for sorting algorithms, addressing the producer-consumer synchronization challenge through multithreading.

Open Lab Final Project, Credits: 1

In this capstone course, students will integrate their learning from the Open Lab curriculum into a project suitable for showcasing in their professional portfolio. The project will focus on implementing a Watchdog project in C, utilizing data structures, multiple threads, and concepts from previous courses including Cyber Kill Chain, Exploitation, Visibility & Monitoring to demonstrate students' proficiency in the subject matter.

Specialization Tracks

After completion of all the Open Lab courses, students select one of the following specialization tracks:

- **RD track**
- **Full Stack track**

The following are fundamentals common to both tracks:

- Advanced system programming
- Object Oriented Programming (OOP)
- Distributed systems
- Networking & network software development
- Design patterns
- Building user-friendly interfaces
- Multi-tier architecture
- Debugging complex systems
- Project architecture & design
- UML
- Testing strategies
- Asynchronous communication
- Client / server
- Event-driven development
- Code refactoring
- S.O.L.I.D. principles
- Strictly layered systems / layered architecture
- Real-world software engineering practices
- Separation of Concerns (SoC)
- Backward compatibility
- Quality-driven development
- Dynamic programming
- Technical presentation

RD Track

The RD track provides a 'bottom-up' approach to learning – with in-depth theory and an array of under-the-hood technologies and respective skills that are relevant for a wide range of software development disciplines. Topics focus on systems, networks, architectures & topologies, environments, frameworks, professional tools, coding styles, development environments, the software development lifecycle (SDLC), additional programming languages and implementation techniques.

Introduction to File System, Credits: 1

This foundational course delves into the essentials of file systems, and their underlying implementations. Students will gain a solid understanding of file system theory and engage in hands-on practice parsing file systems, preparing them for more advanced topics in computer science.

Object Oriented Programming, C++ Internals and Applied C++, Credits: 4

This in-depth course explores the C++ language and Object-Oriented Programming design principles, offering students a comprehensive understanding of C++ internals. Through practical applications, students will master Resource Acquisition Is Initialization (RAII), bitwise operations, standard containers, scope locks, and shared pointers, strengthening their programming skills.

Networking and IPC, Credits: 2

This course introduces students to the core concepts of inter-process communication theory, mechanisms, and the OSI 7-layer model. By examining the inner workings of shared memory, pipes, TCP, UDP, the HTTP & HTTPS protocols, and the cyber security fundamentals of networking, infrastructure & application security, students will gain a broad understanding of various networking and security principles critical to computer science.

Implementations of Design Patterns, Credits: 3

Focusing on software design capabilities, this course teaches students the application of Unified Modeling Language (UML) and design patterns. Students will investigate and implement advanced design patterns, emphasizing multithreading, libraries, scheduling, and factories, equipping them with the skills to develop efficient, flexible, and maintainable software.

Final Project – Infrastructure, Credits: 2

In this capstone course's first segment, students will apply their accumulated skills and secure coding knowledge to a large-scale IoT project. This real-world RAID01-driven application, suitable for inclusion in their professional portfolio, integrates multiprocessing, multithreading, IPC, advanced design patterns, scheduling, and file system management. Students will focus on the master side of the project, ensuring a robust and efficient system.

Final Project – NBD Application and Communication, Credits: 1

The second part of this capstone course involves the continued development of the secure IoT project, with students concentrating on master-side kernel-to-user-mode communication. They will create a system capable of recognizing file system changes in the kernel and reacting accordingly in user mode, broadening the project's functionality and versatility.

Final Project – Embedded, Credits: 3

In the final segment of the capstone course, students will implement the minion side of their secure IoT project, compiling and running it within an embedded architecture. This hands-on experience emphasizes seamless integration and finalization, demonstrating the comprehensive skills and knowledge acquired throughout the RD Track.

Methods and Applications of Interviews, Credits: 1

This course aims to prepare students for professional interviews by emphasizing a strategic approach to both technical and interpersonal skills through guided instruction, interactive exercises, simulation and feedback sessions. Students will learn to identify key interview topics, prioritize information, and effectively communicate within the interview framework, ultimately increasing their chances of success.

Full Stack Track

The Full Stack – Back-End track provides a ‘top-down’ approach to learning, with a comprehensive exposure to the technologies, tools and trends of back-end Web development and Web services, including specialized and best programming practices, multiple types of system development, databases, APIs, languages, UI/UX basics, networks, and cloud computing.

Object Oriented Programming, Java Internals and Applied Java 1, Credits: 2

This foundational course provides students with an introduction to the Java language and its core components, such as setup, usage, garbage collection, and Object-Oriented Programming principles. Students will examine enums and inner classes, practice Object-Oriented Design using UML, and learn to write efficient automated tests with JUnit, building a solid foundation in Java development.

Object Oriented Programming, Java Internals and Applied Java 2, Credits: 2

Building upon the first course, this class delves into advanced Java concepts, including reflection, exceptions, generics, collections, and concurrency. Students will explore the intricacies of Object-Oriented Programming and apply this knowledge to implement various data structures, further strengthening their Java development skills.

Implementations of Design Patterns, Credits: 2

In this course, students will investigate design patterns and their implementation, as well as the design process for addressing common software challenges. Through hands-on learning and practical exercises, students will develop problem-solving skills and a strong understanding of design principles, enabling them to create more robust and efficient software systems.

Applied Backend Development, Credits: 2

This course examines the principles and advanced aspects of Java backend development, focusing on interactions with files, JAR files, networking, REST APIs and the cyber security fundamentals of networking, infrastructure & application security. Students will gain hands-on experience using Apache tools such as Maven and Tomcat, developing a practical understanding of Java's backend capabilities and potential real-world applications.

Applied Databases, Credits: 2

This comprehensive course equips students with a deep understanding of database management systems, covering both relational (MySQL) and non-relational (MongoDB) databases. Students will learn to perform Create, Read, Update, and Delete (CRUD) operations using the Java Database Connectivity (JDBC) API, acquiring the skills needed to effectively work with various database systems in real-world applications.

Introduction to Frontend Development, Credits: 1

This introductory course acquaints students with the fundamentals of frontend web development, providing the foundation needed to create visually appealing and interactive web pages. Students will learn the basics of HTML, CSS, and JavaScript, exploring their interplay and how to leverage each to produce a user-friendly web experience. Practical exercises and projects will help students develop the skills necessary to excel in frontend web development.

Final Project, Credits: 4

In this capstone course, students will apply their knowledge from the Full Stack Track to an intricate, secure IoT database project. This real-world project, suitable for inclusion in a professional portfolio, focuses on the application of backend and frontend technologies for data collection, tracking, bug logging, and updating IoT products. Students will design and implement a comprehensive system, emphasizing scalability, configurability, asynchronous design patterns, and the creation of fault-tolerant servers, ensuring a robust and efficient final product.

Methods and Applications of Interviews, Credits: 1

This course aims to prepare students for professional interviews by emphasizing a strategic approach to both technical and interpersonal skills through guided instruction, interactive exercises, simulation and feedback sessions. Students will learn to identify key interview topics, prioritize information, and effectively communicate within the interview framework, ultimately increasing their chances of success.