Course Syllabus

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>ECTS Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOC-522DL</td>
<td>Smart Contracts Programming</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prerequisites</th>
<th>Department</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOC-512DL</td>
<td>Digital Innovation</td>
<td>Fall/Spring</td>
</tr>
<tr>
<td>BLOC-514DL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of Course</th>
<th>Field</th>
<th>Language of Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elective</td>
<td>Smart Contracts, Programming</td>
<td>English</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level of Course</th>
<th>Lecturer(s)</th>
<th>Year of Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd Cycle</td>
<td>Dr. Klitos Christodoulou</td>
<td>2nd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode of Delivery</th>
<th>Work Placement</th>
<th>Corequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance Learning</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Course Objectives:

This course is designed for developers that have familiarity with other high-level programming languages. The main element of this course is to provide students with a solid understanding of the many opportunities for building decentralized applications using the Web3 stack and the Turing-complete Solidity language over the Ethereum Virtual Machine (EVM).

Learning Outcomes:

After completion of the course students are expected to be able to:

- Understand and evaluate the stack of protocols that will form the future Web 3.0 and its decentralized nature
- Understand and evaluate the components of blockchain-based technologies which support Turing-complete languages
- Explain in detail the architecture of Ethereum and the structure of the Ethereum Virtual Machine (including Byte Code interpretation)
- Establish a deep understanding of the Ethereum model, its consensus model, code execution, operation of its network, storage options and main actors that participate on its protocol
- Understand the inner workings of smart contracts as means for developing decentralized applications;
• Develop smart contracts using the Solidity programming language (including a deep understanding of the provided Libraries)
• Build a local Ethereum Network with Geth, and get familiar with a various development environments (e.g., Truffle, Remix - Ethereum IDE)
• Understand the interaction between the enclosed smart contract network and the external world, be aware of further implications these interactions pose to the aspect of decentralization
• Reuse common implementation patterns, like modifiers and contract driven development;
• understand the smart contract development lifecycle (contract implementation, testing, deploying, and migrating a contract)
• Understand a set of technologies that support the backbone decentralized storage network (e.g., IPFS, Swarm).

Course Content:

• Introduction to Blockchain and Ethereum
  o What is a Blockchain and why should I care?
  o Blockchain Architectural Overview
  o The Web of Trust
  o Ethereum’s main components
  o Ethereum’s sub-protocols
  o The new generation of the Web (i.e., Web3.0)
  o Smart Contracts and Decentralized Applications (dApps)
  o Web apps vs. dApps

• Introduction to Smart Contracts
  o An overview to the history of smart contracts
  o An intro to the life-cycle of a smart contract
  o Ethereum’s smart contract languages
  o Interfacing with Ethereum Networks (overview of Ethereum Networks, Clients, Wallets, Transactions etc.)
  o The Solidity Programming Language
  o Development Environments

• Blockchain technology Supporting Turing-Complete Languages
  o A comparison of Ethereum and Bitcoin
Overview of Ethereum’s tech stack, architecture
- The Ethereum reward scheme, Mist, EVM, Swarm, Whisper, Eth, Gas
- A simple Solidity Contract (Contract Walk-through)
- The Solidity compiler
- Ethereum Contract ABI
- Deployment with the Web3.js or Web3J library

Virtual Machines and Beyond
- History of Virtual Machines
- State replication, consensus and the Ethereum Architecture
- Introduction to the Ethereum Virtual Machine and EVM Byte Code interpretation
- Incentivisation structures, rewards schemes, and gas pricing

Intro to the dApp Development Pipeline
- Introduction to development with Solidity
- Development environments (Truffle)
- Intro to Solidity
- Smart contract layout
- The structure of .sol source file

Deep-dive into Solidity
- Understanding the different compiler versions and pragmas
- Authoring smart contracts
- Contract definitions
- Basic data types
- Local and State Variables

Global Variables and Functions
- Predefined Global Variables
- Structs and Enums
- Mapping and Arrays
- Build-in Functions (e.g., addmod, keccak256)
- User Functions

Expressions and Control Structures
- Valid expressions of the language
• Exception Handling (e.g., assert, require, revert, throw)
  • Events and Logging
  • Conditional logic
  • Implementation of loops

• Object Oriented Constructs
  • Contract constructor and selfdestruct
  • Function Modifiers and Fallback functions
  • Calling other contracts
  • Inheritance and Multiple Inheritance
  • Declaring Abstract Classes and Interfaces
  • Implementation of Abstract interfaces
  • Function Overloading

• Experimenting with Front-end Libraries
  • Intro to front-end web interfaces
  • Decentralized Data Storage
  • The Ethereum Name Services (ENS)

• Unit Testing and Debugging Contracts
  • Estimating Gas Costs
  • Basics of using Truffle for testing
  • Troubleshooting and Debugging
  • Common design patterns
  • Smart Contract Security – overview of attacks on Ethereum smart contracts

• Deployment Considerations and Other Smart Contract Platforms
  • Smart Contracts Quality Assurance
  • Beyond Ethereum
  • Blockchain-as-a-Service (BaaS) and the Dark Market
  • Secure smart contracts with OpenZeppelin
  • Experimenting with Hyperledger Besu
  • Future Outlook and the Road Ahead (e.g., graph-based blockchain protocols, distributed autonomous organizations, quantum secured blockchains etc)
Learning Activities and Teaching Methods:

Lectures, Seminars, Assignments, Live Discussions, Forum Discussions, Practical Projects

Assessment Methods:

Assignments, Final Exam

Required Textbooks / Readings:

<table>
<thead>
<tr>
<th>Title</th>
<th>Author(s)</th>
<th>Publisher</th>
<th>Year</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mastering ethereum: building smart contracts and dapps</td>
<td>Antonopoulos, Andreas M., and Gavin Wood</td>
<td>O'Reilly Media</td>
<td>2018</td>
<td></td>
</tr>
<tr>
<td>The science of the blockchain</td>
<td>Wattenhofer, Roger</td>
<td>CreateSpace Independent Publishing Platform</td>
<td>2016</td>
<td></td>
</tr>
</tbody>
</table>
In International Conference on Principles of Security and Trust, pp. 164-186

**Recommended Textbooks / Readings:**

<table>
<thead>
<tr>
<th>Title</th>
<th>Author(s)</th>
<th>Publisher</th>
<th>Year</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blockchains and databases: A new era in distributed computing</td>
<td>Mohan, C</td>
<td>In 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 1739-1740. IEEE, 2018.</td>
<td>2018</td>
<td></td>
</tr>
</tbody>
</table>

**Selected online readings:**

- A curated list of awesome Solidity resources, libraries, tools and more https://github.com/bkrem/awesome-solidity

**Note:** an updated list of readings in provided at the end of each lecture given the fact that Digital Currency and Blockchain Technologies constitute recent and rapidly evolving disciplines.