Master of Science in Computer Science

Specialization in Software Engineering

Tier 1

Title	ECTS / Workload
Python Programming: Foundations and Best Practices	5 / 125
Mathematics for Computer Science and Introduction to Problem-Solving Techniques	5 / 125
Basic Algorithms and Data Structures	5 / 125

Tier 2

HTML and CSS Fundamentals for User Interface Design	5 / 125
JavaScript Fundamentals: From Basics to Advanced Concepts	5 / 125
Advanced JavaScript and TypeScript: Tools and Best Practices	5 / 125
Relational Databases: Concepts and Techniques	5 / 125
Algorithmic Paradigms and Techniques for Problem Solving	5 / 125
Fullstack. Back End Development: Node.js	5 / 125
Agile Product Management for Software Development Teams	5 / 125
Digital Design for the Web: Principles and Applications (electives)	5 / 125
Interaction Design and Design Thinking for UX/UI (electives)	5 / 125
Mastering Front-End Development with React (electives)	5 / 125
Cross-Platform Mobile App Design and Development with React Native (electives)	5 / 125

FullStack Web Development with Python (electives)	5 / 125
Introduction to software development and quality control (electives)	5 / 125

Tier 3

Title	ECTS / Workload
Foundations of Cloud Computing	5 / 125
DevOps CI/CD	5 / 125
System Design	5 / 125
Career Strategies and Soft Skills for IT Professionals	5 / 125
Applied Computer Science: Capstone Project	10 / 250

Course Descriptions & Intended Learning Outcomes

Python Programming: Foundations and Best Practices

Description: This course provides a practical and detailed understanding of popular programming paradigms and data storage types. Students learning this will be able to write and solve programming problems. The course starts from the basics about functions, various built in functions and how to code user defined functions. Then students will learn about various data type storages and learn about lists and how various manipulations can be done lists like list slicing and also go through examples of 2D Lists.

While learning how to create functions students have to learn how various results and inputs can be stored using different data types. After the introduction and discussion on Lists, students will go through sets, tuples, Dictionaries and Strings.

The student should be well prepared to apply these concepts and build algorithms and software using what they learnt in this course.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for storing data in a computer

program
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to solving problems with 2D lists
- Propose appropriate solutions to complex and changing problems of data storage, programming functions, and algorithms

Mathematics for Computer Science and Introduction to Problem-Solving Techniques

Description: Mathematics and computer science are closely related fields. Problems in computer science are often formalized and solved with mathematical methods. It is likely that many important problems currently facing computer scientists will be solved by researchers skilled in algebra, analysis, combinatorics, logic and/or probability theory, as well as computer science.

This course covers discrete mathematics for computer science and engineering. Topics may include asymptotic notation and growth of functions; permutations and combinations; counting principles; discrete probability. Further selected topics may also be covered, such as recursive definition and structural induction; state machines and invariants; recurrences; generating functions.

Students will be able to explain and apply the basic methods of discrete (noncontinuous) mathematics in computer science. They will be able to use these methods in subsequent courses in the design and analysis of algorithms, computability theory, software engineering, and computer systems. The focus of the course is real-world problems and applications often found in business and industry.

Besides, students will learn about different problem-solving strategies and when to use them will give a good start. Problem solving is a process. Most strategies provide steps that help you identify the problem and choose the best solution.

Building a toolbox of problem-solving strategies will improve problem solving skills. With practice, students will be able to recognize and choose among multiple strategies to find the most appropriate one to solve complex problems. The course will focus on developing problem-solving strategies such as abstraction, modularity, recursion, iteration, bisection, and exhaustive enumeration.

The course will also introduce arrays and some of their real-world applications, such as prefix sum, carry forward, subarrays, and 2-dimensional matrices. Examples will include industry-relevant problems and dive deeply into building their solutions with various approaches, recognizing each's limitations (i.e when to use a data structure and when not to use a data structure).

Key Intended Learning Outcomes:

- Assess, analyse, and criticise the various strategies for evaluating algorithmic cost arising in the context of computational problem-solving and handling matters arising in the context of structured data
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle evaluating algorithmic

performance and solving problems with structured data
- Propose appropriate solutions to complex and changing problems pertaining to problem-solving in software development

Basic Algorithms and Data Structures

Description: This course is aimed to build a strong foundational knowledge of data structures (DS) used extensively in computing. The module starts with introducing time and space complexity notations and estimation for code snippets. This helps students be able to make trade-offs between various Data Structures while solving real world computational problems. The module introduces most widely used basic data structures like Dynamic arrays, multi-dimensional arrays, Lists, Strings, Hash Tables, Binary Trees, Balanced Binary Trees, Priority Queues and Graphs. The module discusses multiple implementation variations for each of the above data-structures along with trade-offs in space and time for each implementation. In this course, students implement these data-structures from scratch to gain a solid understanding of their inner workings. Students are also introduced to how to use the built-in data-structures available in various programming languages/libraries like Python/NumPy/JavaScript. Students solve real-world problems where they must use an optimal DS to solve a computational problem at hand.

Key Intended Learning Outcomes:

- Assess, analyse, and criticise the various strategies for handling matters arising in the context of Data structures
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should implement Data structures
- Propose appropriate solutions to complex and changing problems pertaining to different approach to Data structures applications

HTML and CSS Fundamentals for User Interface Design

Description: This is a hands-on course on designing responsive, modern and light-weight UI for web, mobile and desktop applications using HTML5 and CSS. Throughout the course students will learn how web browsers, mobile apps and web servers work. We then dive into each of the nitty gritty details of HTML5 to build webpages. We would start with simple web pages and then graduate to more complex layouts and features in HTML. We then go on to learn stylesheets based on CSS and how browsers interpret CSS files to render web pages. Once again, we use multiple real world example web pages to learn the internals of CSS. We learn popular good practices on writing responsive HTML and CSS code which is also interoperable on mobile browsers, apps and desktop apps. We would introduce students to building desktop apps using HTML and CSS using appropriate toolkits. We would also study semantic markup, which is an important component of web application development in terms of accessibility and SEO. Students will learn about different types of HTML tags used to describe the structure and content of web pages, allowing browsers and other interpreters to correctly interpret content and improve its readability for people and search engines.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for handling matters arising in the context of Front end UI/UX development
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle Front end UI/UX development
- Propose appropriate solutions to complex and changing problems pertaining to Front end UI/UX development

JavaScript Fundamentals: From Basics to Advanced Concepts

Description: This course is a hands-on course covering JavaScript from basics to advanced concepts in detail using multiple examples. We start with basic programming concepts like variables, control statements, loops, classes and objects. Students also learn basic data-structures like Strings, Arrays and dates. Students also learn to debug our code and handle errors gracefully in code. We learn popular style guides and good coding practices to build readable and reusable code which is also highly performant. We then learn how web browsers execute JavaScript code using V8 engine as an example. We also cover concepts like JIT-compiling which helps JS code to run faster. This is followed by slightly advanced concepts like DOM, Async-functions, Web APIs and Fetch which are very popularly used in modern front end development. We learn how to optimize JavaScript code to run on both mobile apps and mobile browsers along with Desktop browsers and as desktop apps via ElectronJS. Most of this course would be covered via real world examples and by learning from JS code of popular open-source websites and libraries.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for handling matters arising in the context of JavaScript
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle JavaScript
- Propose appropriate solutions to complex and changing problems pertaining to JavaScript

Advanced JavaScript and TypeScript: Tools and Best Practices

Description: This advanced JavaScript course builds on the foundational concepts covered in the JavaScript course, with a focus on more advanced concepts and best practices for building modern, performant web applications. Through hands-on practice and real-world examples, students will learn how to optimize JavaScript code for mobile and desktop devices, work with the DOM and Web APIs, and interact with backend APIs.

The course will begin with an overview of event propagation and optimization techniques, including event bubbling, delegation, and throttling. Students will also learn about lazy loading images, using libraries via CDN, and other performance optimization techniques.

Next, the course will cover project infrastructure and web storage, including working with Node.js, npm package management, code modularity, and syntax for ECMAScript modules. Students will learn about Webpack, Babel, and other tools for transpiling and bundling code, as well as code formatting and checking best practices.

The course will also cover asynchrony and date handling in JavaScript, with a focus on the Promise API, async/await syntax, and event loop. Students will learn how to interact with backend APIs, including working with REST APIs, HTTP methods, headers, and response status codes. They will also learn about pagination techniques, including "load more" buttons and infinite scrolling. Finally, the course will cover CRUD operations with asynchronous functions, including working with private APIs and error handling best practices.

Key Intended Learning Outcomes:

- Analyze and optimize JavaScript code for mobile and desktop devices, using best practices for performance optimization
- Create modular, reusable code using ECMAScript modules and other tools for transpiling and bundling code
- Interact with backend APIs using REST APIs, HTTP methods, and pagination techniques
- Develop asynchronous functions and handle errors effectively for CRUD operations

Relational Databases: Concepts and Techniques

Description: This is a core and foundational course which aims to equip the student with the ability to model, design, implement and query relational database systems for real-world data storage & processing needs. Students would start with diagrammatic tools (ER-diagram) to map a real world data storage problem into entities, relationships and keys. Then, they learn to translate the ER-diagram into a relational model with tables. SQL is then introduced as a de facto tool to create, modify, append, delete, query and manipulate data in a relational database. Due to SQL's popularity, the course spends considerable time building the ability to write optimized and complex queries for various data manipulation tasks. The module exposes students to various real world SQL examples to build solid practical knowledge. Students then move on to understanding various trade-offs in modern relational databases like the ones between storage space and latency. Designing a database would need a solid understanding of normal forms to minimize data duplication, indexing for speedup and flattening tables to avoid complex joins in low-latency environments. These real-world database design strategies are discussed with practical examples from various domains. Most of this course uses the open source MySQL database and cloud-hosted relational databases (like Amazon RDS) to help students apply the concepts learned on real databases via assignments.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for handling matters arising in the context of Relational Databases

- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle Relational Databases
- Propose appropriate solutions to complex and changing problems pertaining to Relational Databases

Algorithmic Paradigms and Techniques for Problem Solving

Description: This is a foundational and mandatory course which aims to build student's ability to apply various algorithmic design methods to provide an optimal solution to computational problems. This course starts with time and space complexity analysis of divide and conquer algorithms using recursion-tree based methods and Master's theorem. Students would also learn about amortized time and space complexity analysis for randomized/probabilistic algorithms. Various algorithmic design strategies would be introduced via real world examples and problems. Students would learn when, where and how to optimally use Divide and Conquer, Dynamic programming (top-down and button-up), Greedy, Backtracking and Randomization strategies with examples. The module uses various practical examples from Array manipulations, Sorting, Searching, String manipulations, Tree & Graphs traversals, Graph path-finding, Spanning Trees etc., to introduce the above algorithmic strategies in action. Students would implement many of the above algorithmic design methods from scratch as part of the assignments. The module also introduces how some of these popular algorithms are readily available via popular libraries in various programming languages.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for handling matters arising in the context of design and analysis of algorithms
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle algorithmic design methods
- Propose appropriate solutions to complex and changing problems pertaining to design and analysis of algorithms

Fullstack. Back End Development: Node.js

Description: This is a foundational course on building server-side (or backend) applications using popular JavaScript runtime environments like Node.js. Students will learn event driven programming for building scalable backend for web applications. The module teaches various aspects of Node.js like setup, package manager, client-server programming and connecting to various databases and REST APIs. Most of these concepts would be covered in a hands-on manner with real world examples and applications built from scratch using Node.js on Linux servers. This course also provides an introduction to Linux server administration and scripting with special focus on web-development and networking. Students learn to use Linux monitoring tools (like Monit) to track the health of the servers. The module also provides an introduction to Express.js which is a popular light-weight framework for Node.js applications. Given the practical nature of this course, this would involve building actual website backends via assignments/projects for ecommerce, online learning and/or photo-sharing.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for handling matters arising in the context of Back End Development
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle Back Eend Development
- Propose appropriate solutions to complex and changing problems pertaining to Back End Development

Agile Product Management for Software Development Teams

Description: Every organization is building products to solve the pain points of its customers. Product managers are a critical part of an organization, who make sure that evolving customer needs, and market trends are observed and converted into delightful solutions which help businesses get its outcomes.

In this course, students will get a fundamental understanding of product management practices.

This will give them a comprehensive view of the complete product management life cycle.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for improving a product after launch
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to measuring user engagement
- Propose appropriate solutions to complex and changing problems of product success or failure in real-world engineering and science contexts

Digital Design for the Web: Principles and Applications

Description: This web design course is designed to provide students with the skills and knowledge necessary to create attractive, functional, and effective websites, including landing pages and company websites. The course covers a range of topics, including the fundamentals of web design such as finding references, researching competitors, basic research, wireframing, prototyping, grids, composition, typography, color, raster and vector graphics, user interface patterns, and adaptation.

Students will learn the basic laws of UX and the main user behavior patterns on the website. Students will be introduced to tools such as Figma, FigJam, Protopie, which will be used to create wireframes, layouts, and prototypes. The course will also include preparation of a case for publication on Behance, which will provide an opportunity to demonstrate skills to employers.

Key Intended Learning Outcomes:

- Demonstrate proficiency in using Figma to create wireframes, prototypes, and high-fidelity designs.
- Analyze and evaluate different web design principles, including wireframing, prototyping, composition, typography, color, and graphics, to create functional and visually attractive websites.
- Apply critical thinking and problem-solving skills to analyze and address web design-related issues and effectively communicate solutions to clients and stakeholders.

Interaction Design and Design Thinking for UX/UI

Description: User Experience and User Interface (UX/UI) design is about understanding user needs and preferences, and creating digital products that meet those needs. Throughout this course, students will learn the fundamental skills and tools necessary to develop an effective user interface and experience.

Students will learn about the design thinking process, user personas and flows, customer journey mapping, and data visualization. They will also learn about the importance of collaboration between designers and developers, as well as how to test and iterate design.

The course covers essential topics such as Figma Pro, design system creation, mobile-first design, smart animation, and microcopy. Students will learn the process of designing from ideation to prototype creation, testing, and improvement, and understand how to work through iterations. The course includes an understanding of UX testing and its types, and working with analytics.

By the end of the course, students will have a clear understanding of how to create digital products that are aesthetically appealing and convenient for the user.

Key Intended Learning Outcomes:

- Assess and analyze user needs and preferences for the development of effective user interface and experience
- Develop ways to visualize data to create attractive and informative digital products
- Gain a deep understanding of the design thinking process and its application in solving complex design problems
- Create and iterate designs through prototyping and user testing, ensuring the final product meets user needs and desires.

Mastering Front-End Development with React

Description: This course builds upon the introductory JavaScript course to acquaint students of popular and modern frameworks to build the front end. We focus on one of the most popular and advanced frameworks/libraries in use – React.js. Students learn various components and data flow to learn to architect real world front end using React.js. This would be achieved via multiple code examples and code-walkthroughs from scratch. This helps students to build applications for various platforms using only JavaScript.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for handling matters arising in the context of front end development
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle front end development applications
- Propose appropriate solutions to complex and changing problems pertaining to front end development

Cross-Platform Mobile App Design and Development with React Native

Description: Mobile app design is a rapidly developing field that requires a deep understanding of user needs, technology, and UX design principles. This course aims to provide students with an in-depth understanding of various aspects involved in designing and developing cross-platform mobile applications using React Native. The course covers a wide range of topics, including React Native architecture, UI components, navigation, data management, user engagement, animation, and app store optimization.

Students will learn about the unique features of mobile app design, types of apps and technologies used in this field. The course emphasizes the importance of cross-platform compatibility, ensuring that the mobile apps created can run seamlessly on both iOS and Android platforms. The course will also cover familiarity with key design patterns for mobile apps, user engagement, animation, and preparing the app for publication.

Throughout the course, students will have the opportunity to work on real-world projects and assignments, allowing them to apply their learning to practical situations. They will learn how to analyze and evaluate different types of mobile apps and technologies used in mobile app design, as well as how to apply design principles and design patterns to create mobile app interfaces that are user-friendly and engaging.

In addition, the course covers important topics such as app store submission process and optimizing app performance, enabling students to prepare their mobile apps for publication.

Key Intended Learning Outcomes:

- Understand the principles of cross-platform mobile app design and development with React Native.
- Analyze and evaluate mobile app design features, types of apps, and technologies used in mobile app design.
- Apply UX design principles and patterns to create user-friendly and attractive

interfaces for mobile apps using the React Native framework.
- Acquire skills to prepare the app for publication, including understanding the process of submitting to app stores and optimizing performance.

FullStack Web Development with Python

Description: Advanced Python Programming builds on introductory programming courses to illustrate object-oriented programming concepts, database design in Python, and the basics of Machine Learning with Python libraries. Students will learn how to solve problems in Python, develop design patterns in Python code, develop internet applications with Python, and collaborate with other students to implement projects. The course introduces advanced features such as decorators and generators, as well as a thorough exploration of the Python development environment.

This course is designed to prepare students for an entry-level developer position.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for implementing internet applications in Python
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to developing code for machine learning
- Propose appropriate solutions to complex and changing problems pertaining to mathematical analysis of data in real-world engineering and science contexts

Introduction to software development and quality control

Description: This course is designed to provide a comprehensive understanding of Quality Assurance (QA) in software development. The course will cover the fundamental principles of testing and the different types of testing that are conducted at various levels of the software development life cycle. Students will also learn about the different testing techniques used in QA, such as black box, white box, and experience-based testing.

The course will also introduce students to various testing tools and methodologies that are commonly used in industry, including test management tools, SQL databases, Postman, and mobile testing. Students will learn about web technologies and the client-server architecture, as well as front-end and back-end development. The course will cover the basics of HTML/CSS, modern application architecture, and working with command-line tools like CI/CD and Git.

Throughout the course, students will develop a solid understanding of QA and its role in software development. They will learn how to develop test documentation and will gain practical experience in implementing various testing strategies. They will also learn how to analyze and critique different QA methodologies and propose appropriate solutions to complex and changing problems in the context of data structures. Students will be able to apply their understanding of web technologies and modern application architecture to design and test web applications, and will be well-equipped to pursue careers in software development or QA.

Key Intended Learning Outcomes:

- Analyze, evaluate, and apply various testing methodologies and techniques in the context of QA
- Develop and implement effective test documentation for software development projects
- Critique and recommend appropriate QA strategies for complex and changing problems
- Utilize various testing tools and technologies to design, implement, and manage QA processes
- Apply understanding of web technologies and modern application architecture to design and test web applications.

Foundations of Cloud Computing

Description: This is a course that focuses both on architectural design and practical hands-on learning of the most used cloud services. The module extensively uses Amazon Web services (AWS) to show real world code examples of various cloud services. It also covers the core concepts and architectures in a platform agnostic manner so that students can easily translate these learnings to other cloud platforms (like Azure, GCP etc.). The module starts with virtualization and how virtualized compute instances are created and configured. Students also learn how to auto-scale applications using load balancers and build fault tolerant applications across a geographically distributed cloud. As relational databases are widely used in most enterprises, students learn how to migrate and scale (both vertically and horizontally) these databases on the cloud while ensuring enterprise grade security. Virtual private clouds enable us to create a logically isolated virtual network of compute resources. Students learn to set up a VPC using virtualized-compute-servers on AWS. The course also covers the basics of networking while setting up a VPC. Students learn of the architecture and practical aspects of distributed object storage and how it enables low latency and high availability data storage on the cloud.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for handling matters arising in the context of cloud computing
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle cloud computing
- Propose appropriate solutions to complex and changing problems pertaining to cloud computing

DevOps CI/CD

Description: This course provides students with hands-on experience on deploying high velocity applications and services reliably on complex and distributed infrastructure. DevOps as a philosophy is a key driver of the modern software life cycle which prefers rapid and reliable delivery of functionality and features via code. We start with a solid introduction to Linux scripting and networking. Then, we learn popular methodologies to deploy complex and distributed software like microservices, containerization (Docker) and orchestration (Kubernetes). All of this would be introduced with real world examples from the industry. We also focus on Continuous Integration and Continuous Delivery (CI/CD) methodology and how it can be achieved using popular toolchains like Jenkins. We dive into how automated testing of software can be achieved using libraries like Selenium. This shall be followed by more advanced techniques like serverless-compute, Platform as a service model and Cloud-DevOps. Students would learn to monitor and log key data points to ensure they maintain a healthy system and adapt it as needed. Infrastructure-as-code is a key component of modern DevOps especially on cloud and containerized applications which would also be covered with real-world examples.

Key Intended Learning Outcomes:

- Assess, analyse, and criticise the various strategies for handling matters arising in the context of DevOps
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle DevOps
- Propose appropriate solutions to complex and changing problems pertaining to DevOps

System Design

Description: This course is aimed at equipping students with skills to architect the high level design (a.k.a. system design) of software and data systems. We start with some of the good to have properties of large complex software systems like scalability, reliability, availability, consistency etc. The module teaches various patterns and design choices we have to satisfy each of these good to have properties. We then go on to understand key components of system design like load-balancers, microservices, reverse-proxies, content-delivery networks etc. Students learn how each of them work internally along with real world implementations of each. We study various NoSQL data stores, their internal architectures and where to use which one with real-world examples. Students also learn popular data encoding schemes like XML and JSON. We learn how to build data pipelines using batch and stream processing systems. We also work on multiple real world cases on architecting on the cloud using popular open-source libraries and tools. Students will study design documents and high-level-design of popular internet applications and services like video-conferencing, recommender-systems, peer-to-peer chat, voice-assistants etc.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for handling matters arising in the context of System Design
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to how managers should handle System Design
- Propose appropriate solutions to complex and changing problems pertaining to System Design

Career Strategies and Soft Skills for IT Professionals

Description: This course is designed to equip IT professionals with the soft skills and career strategies required for success in the technology industry. The course is project-based and covers a range of topics such as communication skills, teamwork, time management, leadership, networking, and career development.

The course covers the entire lifecycle of a technology project, from requirement gathering to delivery and maintenance. Students will learn how to communicate effectively with stakeholders, manage their time efficiently, lead a team, and collaborate effectively in a team environment.

The course also covers aspects of career development, such as networking and building professional relationships, creating a personal brand, and developing a career plan. Students will learn how to identify their strengths and weaknesses, and how to leverage their skills and experience to advance their careers in the technology industry.

Key Intended Learning Outcomes:

- Develop and demonstrate effective communication skills.
- Collaborate effectively in a team environment.
- Develop and demonstrate leadership skills.
- Build and maintain professional relationships.
- Develop and execute a career plan.

Applied Computer Science: Capstone project

Description: This is a project-based course, with the aim of building the required skills for creating web-based software systems. The course covers the entire lifecycle of building software projects, from requirement gathering and scope definition from a product document, to designing the architecture of the system, and all the way to delivery and maintenance of the software system.

The course covers both frontend, which is, building browser-based interfaces for users, using frontend web frameworks, and also building the backend, which is the server running an API to serve the information to the frontend, and running on an SQL or similar database management system for storage.

All aspects of delivering a software project, including security, user authentication and authorisation, monitoring and analytics, and maintaining the project are covered. The course also covers the aspects of project maintenance, like using a version control system, setting up continuous integration and deployment pipelines and bug trackers.

Key Intended Learning Outcomes:

- Assess, analyze, and criticize the various strategies for orchestrating and deploying a software to the cloud
- Compare and evaluate the different methodologies recommended in scholarly sources pertaining to data schema design

- Propose appropriate solutions to complex and changing problems of software maintenance in real-world engineering and science contexts