Masters Program in Computer Science - Course Descriptions

Below is a listing of courses offered in the MPCS. For updated course offerings, descriptions and prerequisites for each year, please visit our online course schedule at https://mpcs-courses.cs.uchicago.edu/2019-20/courses.

**Immersion Courses / Prerequisites**

**Concepts of Programming (MPCS 50101)**
In this course students will get an introduction to the field of computer science by learning to program in Java. Students will write roughly two or three programs of significance each week to learn foundational programming principles and practices for writing clean, readable code, and learning how think and solve problems like a computer scientist. Along with basic principles like procedural abstraction, recursion, and handling input and output, an emphasis will be placed on theories and principles of Object Oriented software design, analyzing algorithms and choosing appropriate data structures to solve problems.

**Math for Computer Science: Discrete Math (MPCS 50103)**
This course in an introduction to discrete mathematics oriented toward computer science. The course emphasizes mathematical proof and problem solving, employed on a variety of useful topics: logic; proof by induction; counting, factorials, and binomial coefficients; discrete probability; random variables, expected value, and variance; recurrences; graphs and trees; basic number theory; asymptotic notation, and rates of growth. On completion of the course, students will have been trained to think about and absorb mathematical concepts, to solve problems requiring more than standard recipes, and express mathematical notions precisely. They will be able to use ideas and techniques from discrete mathematics in subsequent courses in computer science, in particular courses in the design and analysis of algorithms, networks, numerical methods, software engineering, data analysis, and data mining.

**Core Courses**

**Advanced Algorithms (MPCS 55005)**
The course is a second course on the design and analysis of efficient algorithms, with emphasis on developing techniques for the design and rigorous analysis of algorithms rather than on implementation. Emphasis is placed on fundamental algorithms and advanced methods of algorithmic design. Techniques to be covered include network flow, dynamic programming, linear programming, randomization, and approximation algorithms. NP-complete problems and reductions will also be studied. Students who complete the course will have increased familiarity with many of the techniques that apply in the design of efficient algorithms and some acquaintance with problems known to be NP-complete.

**Advanced Databases (MPCS 53003)**
The objective of this course will be to (i) expand the knowledge by covering new topics that represent the state-of-the-art in database management systems and distributed systems, and (ii) to build upon foundations developed in MPCS 53001 - Databases by covering topics in greater
depth.

**Advanced Programming (MPCS 51100)**
Advanced Programming fulfills the MPCS Core Programming requirement, but is intended for students who are joining the program with an existing degree in Computer Science, or with substantial experience in programming. This course will be taught primarily in C, including an accelerated introduction to the C language for students who have not used C before. The course will cover advanced data structures and topics in concurrent and multicore programming not covered in the Java Programming or C Programming courses.

**Algorithms (MPCS 55001)**
The course is an introduction to the design and analysis of efficient algorithms, with emphasis on developing techniques for the design and rigorous analysis of algorithms rather than on implementation. Algorithmic problems include sorting and searching, discrete optimization, and algorithmic graph theory. Design techniques include divide-and-conquer methods, dynamic programming, greedy methods, graph search, as well as the design of efficient data structures. Methods of algorithm analysis include asymptotic notation, evaluation of recurrences, and the concepts of polynomial-time algorithms. NP-completeness is introduced toward the end of the course. Students who complete the course will have demonstrated the ability to use divide-and-conquer methods, dynamic programming methods, and greedy methods, when an algorithmic design problem calls for such a method. They will have learned the design strategies employed by the major sorting algorithms and the major graph algorithms, and will have demonstrated the ability to use these design strategies or modify such algorithms to solve algorithm problems when appropriate. They will have derived and solved recurrences describing the performance of divide-and-conquer algorithms, have analyzed the time and space complexity of dynamic programming algorithms, and have analyzed the efficiency of the major graph algorithms, using asymptotic analysis.

**C Programming (MPCS 51040)**
This is an accelerated introduction to the C (not C++) Programming Language designed for students with prior programming experience. C is in many ways the lingua franca of computing, and a broad range of programming languages and related technologies derive from the basic principles of C memory management, control flow, and abstraction. Though there are many subtleties, C is not a big language, and it is expected that students will leave the course with a relatively deep understanding of the key concepts, which will then form a solid foundation for studying higher-level technologies. At the same time, C itself remains a very practical language, particularly so in areas such as scientific programming, high-performance computing, application level library design, systems programming, network programming, multi-threaded programming, etc. Students who successfully complete the course will be well prepared for subsequent MPCS courses in these areas. The course studies both fundamental and advanced C language constructs in the abstract and reinforces them through a range of exercises in the design of basic and advanced data structures, rudimentary algorithms, and API design.

**Compilers (MPCS 51300)**
This class teaches the theory and practice of how to write a compiler, including lexical analysis, grammars, lexers and parsers, type checking, and code generation. For decades, compilers have been the most dynamic and challenging branch in computer science. The main part of this class will focus on providing the basics of the different phases of compilation. Through the course, students will develop appreciation for the implementation strategies behind making an efficient and robust compiler.

The objective of this class is to learn:
- the implementation of a real compiler from scratch
- the answers of how programming language works
- the technical challenges that compilers face
- the basics of reaching performance

**Computer Architecture (MPCS 52010)**

Computer architecture is the science and art of selecting and interconnecting hardware components to create a computer that meets functional, performance and cost goals to run software. With the recent switch from uniprocessor to multicore microprocessors, it has become more important for the programmer to understand the hardware which will run their software. For programs to run fast now, they must become parallel. What this means to the programmer varies depending if they are dealing with an imbedded system, mobile smart phones, laptops or cloud servers. This class will introduce students to the architectural knowledge they need to write high performance software for modern systems.

The class will start with a review of modern computer architecture using a freshly updated classic book, Patterson and Henessy’s Computer Organization and Design. There will be written homeworks the first few weeks and a midterm. This material will give the programmer and system architect a through grounding in modern computer architecture.

The second part of the class will focus on creating a cloud driven distributed microcontroller project. We will use the wi-fi enabled Electric Imp (https://electricimp.com) microcontroller. Students will write cloud server software and microcontroller software which will interact to create a distributed system for the management of local devices over wi-fi and the Internet. Since the security of the Internet of Things is looking as bad as everyone is expecting, we will also discuss the special security issues associated with such distributed systems.

**Databases (MPCS 53001)**

Students will learn database design and development and will build a simple but complete web application powered by a relational database. We start by showing how to model relational databases using the prevailing technique for conceptual modeling -- Entity-Relationship Diagrams (ERD). Concepts covered include entity sets and relationships, entity key as a unique identifier for each object in an entity set, one-one, many-one, and many-many relationships as well as translational rules from conceptual modeling (ERD) to relational table definitions. We also examine the relational model and functional dependencies and their application to the methods for improving database design: normal forms and normalization. After design and modeling, students will learn the universal language of relational databases: SQL (Structured Query Language). We start by introducing relational algebra -- the theoretical foundation of
SQL. Then we examine in detail the two aspects of SQL: data definition language (DDL) and the data manipulation language (DML). Concepts covered include subqueries (correlated and uncorrelated), aggregation, various types of joins including outer joins and syntax alternatives. Students will gain significant experience with writing and reading SQL queries throughout the course in the detailed discussions in class, online homework, and the real-world individual project.

**Distributed Systems (MPCS 52040)**
This class teaches the theory and practice of how to design a distributed system. Cloud computing, drive services, online collaborative working environment, massively multiplayer online gaming, airline reservation systems, e-commerce are examples of distributed systems. Because of a continuous growing number of independent and diverse connected devices, the need of managing heterogeneous architecture became prominent. We will learn how to create a reliable system being easy to use but handling the complexity of having all these machines work collectively.

The objective of this class is to learn distributed system by studying:
- Communication mechanisms
- Synchronization
- Scalability
- Consistency / Replication
- Fault tolerance

**Introduction to Computer Systems (MPCS 52011)**
This course is all about constructing your own knowledge of computer systems by building a general-purpose computer system from the ground up. The objective is to integrate key ideas from algorithms, computer architecture, operating systems, compilers, and software engineering into one unified framework. Along the way, we'll explore ideas and techniques used in the design of modern hardware and software systems, and discuss major trade-offs and future trends. Throughout this journey, you'll gain lots of cross-section views of the field of computer science, from the bare-bone details of switching circuits to the high-level abstraction of object-based software design. By the end of the course, you will have written a computer game in an object-oriented programming language; compiled that program into machine language using the compiler, the virtual machine language translator, and the assembler that you wrote; and run your program on (virtual) hardware that you designed.

**Java Programming (MPCS 51036)**
This is a fast-paced first course in Java for students with some prior programming experience, though not necessarily Java or any other object-oriented language. A strong emphasis will be placed on understanding basic fundamentals of OO design--inheritance, polymorphism, composition, etc. and more generally on applying sound principles of contemporary software engineering and tools to real-world problems. In the latter half of the course, we will cover threads, OO design patterns, as well as certain Java libraries such as Swing. For their final projects, students will develop a multi-threaded, arcade-style game. The course format is both lecture and lab. We will use be using git to facilitate our learning and to manage our projects.
the end of the quarter, students will have a working knowledge of git and know how to manage both local and remote repositories.

**Networks (MPCS 54001)**

Broadly, this course will focus on the history, theory and implementation of computer networks. We will discuss the low-level technologies that move bits around (such as Ethernet and WiFi), the high-level applications that are part of our everyday 21st-century lives (such as email, the Web, and mobile phones), and everything in between (security, TCP/IP). At the completion of this quarter, you will (or should!) be able to explain, in detail, how data makes it way around the Internet when you click on a web link, how you can drive around at 80 MPH talking on a cell phone without the call dropping, how you can make a streaming video call over a lousy wireless link without frame dropping or jitter. In short, we'll pull back the curtain on what can be a somewhat mysterious and magical part of working with computers.

**Operating Systems (MPCS 52030)**

This is an introductory course on operating systems. Students will learn the fundamentals of how modern operating systems are built, from the interface with hardware up through the kernel-userspace boundary. Important topics include the relationship between processes and threads, synchronization, inter-process communication, memory management, file systems, scheduling, I/O, virtualization. These concepts will be reinforced through several large-scale programming projects (in C++), whereby students will implement various sub-components of a real operating system. Prior experience with C and/or C++ required. As appropriate, we’ll use the Linux operating system (written in C) as an example of operating systems design. As time permits, we will also delve into current hot topics in the field (such as multi-core systems, security, and cluster/grid computing).

**Swift Programming (MPCS 51043)**

In this course, students will get an immersive introduction to the field of computer science by learning to program in Swift. Students will learn about fundamental data structures and algorithms, professional coding practices, algorithm design, automated testing, and the fundamentals of object-oriented programming.

While the Swift programming language is a cross-platform, versatile programming language, this class is especially suited for students in the mobile application specialization who are planning on developing for Apple Computer platforms.

**Elective Courses**

**Advanced C++ (MPCS 51045)**

In this continuation of the MPCS 51044 course, we go beyond the basics to cover the powerful and surprising techniques that C++ experts use to write libraries that simultaneously provide the optimum in ease-of-use, abstraction, and performance. If you use C++ in your daily life, you and your team will see substantial benefits from understanding and using C++ at a deeper level.

**Advanced iOS Application Development (MPCS 51032)**
Advances in mobile technologies are changing the way that individuals and businesses use computing devices. This course will explore real-world issues with developing robust, high-performance iOS applications for iPhone, iPod Touch and iPad. The course will consist of lectures, hands-on coding exercises and discussion. Weekly programming assignments will be used to create a portfolio of applications using advanced iOS frameworks and tools, such as Xcode, Interface Builder and Instruments. Throughout the course, students will design and develop an application as a final project. Students may opt to work in collaboration with local companies or emerging start-ups for their project. These opportunities will be discussed during the first week of class and may vary by quarter.

**Android Application Development (MPCS 51031)**
After a quick introduction to mobile computing, competing platforms, Android architecture, market projections, and social and economic implications, we will dive directly into developing several reference implementations. Alternating between theory and practice, and progressing cumulatively, will cover every major feature of the Android platform, including; audio, graphics, internet connectivity, wifi, mapping/geo-positioning, notifications, sms, structured feeds, persistence, threads, states, and inter-process communication, among others. Students will choose a final project, then envision, design, develop, test, and deploy an application to the Android marketplace.

**Applied Software Engineering II (MPCS 51221)**
This course is an intermediate approach to applied software design and development methods for use in creating efficient, reusable, and modular software. This course is offered annually but content and focus change from year to year. Methods we investigate include: classes, inheritance, and polymorphism; design patterns; advanced programming techniques using microservices, event-driven architecture, Hybrid Transactional/Analytical Processing; software frameworks and container-based software development; and advanced techniques including multi-threading. A heavy focus is on design and creativity and what constitutes creative design.

**Advanced Web Development (MPCS 52554)**
This course builds upon MPCS-52553 to enable students to gain mastery over modern web architectures, web services, and visual design.

**Bioinformatics for Computer Scientists (MPCS 56420)**
This course aims to introduce computer scientists to the field of bioinformatics. The vast amounts of data produced in genomics related research has significantly transformed the role of biological research. High-throughput automated biological experiments require advanced algorithms, implemented in high-performance computing systems, to interpret their results. We will focus on analyzing complex data sets in the context of biological problems. Students will design and implement systems that are reliable, capable of handling huge amounts of data, and utilize best practices in interface and usability design to accomplish common bioinformatics related problems. While this course should be of interest for students interested in biological sciences and biotechnology, techniques and approaches taught will be applicable to other fields. This course will present a practical, hands-on approach to the field of bioinformatics. The topics covered in this course will include: software, data mining, high-performance computing,
mathematical models and other areas of computer science that play an important role in bioinformatics. Existing methods for analyzing genomes, sequences and protein structures will be explored, as well as computing infrastructure that support their efficient utilization. Students will be introduced to all of the biology necessary to understand the applications of bioinformatics algorithms and software taught in this course.

**C++ for Advanced Programmers (MPCS 51044)**

This course covers the major features of C++ in an accelerated fashion suitable both for experienced C++ programmers and programmers who are new to C++ as described in the prerequisites below. The course teaches how to get the most out of the current C++11 language, which Bjarne Stroustrup, the inventor of C++, says "feels like a new language." It also discusses how to workaround in old versions of C++. A dominant theme of the course is how to use the unique features of C++ to operate at a high-level of abstraction to support powerful design idioms and improve maintainability while also achieving the kind of performance and low-level control usually associated with lower-level languages such as C and even assembler language.

**Cloud Computing (MPCS 51083)**

Cloud computing is being widely adopted by enterprises of all sizes due to the low initial investment required, attractive operating costs, and elastic capacity that can best serve the highly variable demands of modern applications. Software engineers must be familiar with cloud computing technologies since many new applications they develop will be deployed “in the cloud”, and existing applications will often require integration with cloud-hosted services to take advantage of new capabilities. This course provides an introduction to cloud computing with specific consideration for application development in two contexts: highly scalable (or so-called “web-scale”) web applications, and enterprise applications in a hybrid environment comprising both on-premises and cloud infrastructure. We will focus primarily on infrastructure and platform services, and will introduce software-as-a-service from the perspective of a consuming application. The course will emphasize practical applications of cloud computing technologies, with sufficient exploration of their theoretical underpinnings to inform architectural, design, and implementation decisions.

**Computer and Network Security (MPCS 56515)**

The objective of this course is to provide a basic understanding of Information Technology security - and to build an understanding of the elements that should be in place for an IT environment to achieve an adequate security level. We will begin with a general overview of IT security and introduce a framework for addressing security needs across an enterprise. Major security objectives and technical mechanisms for attaining these objectives will be discussed, including cryptography, authentication systems, Public Key Infrastructure, and platform and network security mechanisms. This course will give an overview of the technical details involved in the platform and network levels of security, including hands-on usage of current tools used in the field. We will look at common TCP/IP applications and discuss their security vulnerabilities. The course material will be presented in a framework of understanding overall risks and how to address them. There will be a great deal of reading in this course. Students should have the ability to read and write in clear prose. Students in this course will be writing an in-depth paper or a project and should have the ability to write a substantial paper.
Entrepreneurship in Technology (MPCS 51250)  
The core theme for the Entrepreneurship in Technology course is that computer science students need exposure to the broad challenges of capturing opportunities and creating companies. Most of the skills required for this process have nothing to do with one’s technical capacity. We’ll explore creating a story, pitching the idea, raising money, hiring, marketing, selling, and more. Real-world examples, case-studies, and lessons-learned will be blended with fundamental concepts and principles. The course will involve a business plan, case-studies, and supplemental reading to provide students with significant insights into the resolve required to take an idea to market. Class discussion will also be a key part of the student experience.

Foundations of Computational Data Analysis (MPCS 53110)  
The course covers statistical methods for exploring, summarizing, and visualizing data sets, for modeling data using probability distributions, for making inferences about a population from samples, for testing hypotheses related to such inferences, and for describing relationships using linear and logistic regressions. It then examines in detail techniques from machine learning used for solving fundamental problems in data mining: classifying data through decision trees, nearest-neighbors, and Bayesian techniques; clustering data through k-means, hierarchical approaches, and density-based techniques; and performing association analysis through the Apriori algorithm. Students use Python for implementing algorithms and Python libraries such as NumPy, SciPy, matplotlib, and pandas for analyzing and visualizing datasets.

High Performance Computing (MPCS 51087)  
Parallel computing allows multiple processing units to work together simultaneously on a common task. For certain types of applications, parallelization can increase execution time in proportion to the number of computers or processors used. This is a huge advantage for applications which have performance and/or memory bottlenecks, such as one typically encounters in financial modelling, physics, engineering, or other applied science domains. This is a fast-paced applied programming course aimed at students with significant development experience in either C, C++, or FORTRAN (Java, Matlab, or Python are also possible, but not ideal). No prior knowledge of parallel computing is assumed. Students should, however, have both an interest and some previous experience in either algorithmic development, numerical methods, applied mathematics, or perhaps any physics or engineering-type discipline. A brief overview of parallel computing will be presented at the outset, but the course will be less on overview of HPC architectures and much more a focus on algorithmic implementation and performance tuning. The goal of the course it to give students experience in developing efficient, scalable (distributed memory) parallel algorithms appropriate for any system running an implementation of the Message Passing Interface (MPI). Assignments will be designed with some flexibility to allow students to explore applying parallel techniques to applications in their own field of interest.

Introduction to Blockchain (MPCS 56600)  
This course is a comprehensive technical introduction to relevant topics in the wider ecosystem surrounding blockchain. Our technological focus will include substantive topics in fundamental
problems that blockchain is attempting to solve (and is generating), including algorithms, cryptography, security and trust, autopoietic peer-to-peer networking, distributed ledgers, double spending, proof of work and ownership issues, decentralized applications, smart contracts, and supporting technologies. With that said, this is not a course in economics or monetary theory, trading cryptocurrencies, nor is it a course on regulatory or legal issues surrounding blockchain, although we will touch on many of these topics throughout the course. We will also cover broader applications of blockchain technology beyond cryptocurrencies and ICOs including use cases from finance, insurance, science, healthcare, pharmaceuticals.

**Introduction to Software Engineering (MPCS 51200)**
Writing first-class software requires top-notch architecture, design and coding skills, but successful software project execution—from identifying the need to providing support—depends on many factors besides technical prowess. This course surveys the key practices and processes that help ensure successful projects. It provides an introduction to central activities of software engineering other than just coding, such as planning, requirements, testing and management. It balances this discussion of typical engineering activities against the development process models in which they take place—specifically, it addresses the tension between traditional plan-driven approaches and adaptive agile techniques. By examining the underlying principles of major development models, it shows how those principles address (or fail to address) the various problems encountered by project teams. Students who complete this course will gain a solid understanding of both plan-driven and agile software development principles and how to negotiate between them in different contexts.

**iOS Application Development (MPCS 51030)**
Advances in mobile technologies are changing the way that individuals and businesses use computing devices. This course will instruct students on the fundamentals of mobile application development using Apple's iOS SDK. An introduction to the Objective-C programming language, including memory management, object-oriented design, and the model-view-controller pattern, will be covered. Using iOS APIs and tools, such as Xcode, Interface Builder and Instruments, students will be able to create fully-featured iPod Touch, iPhone, and iPad applications. User interface and application design considerations specific to mobile technologies will also be explored. The course will consist of lectures, hands-on coding exercises and discussion. Weekly programming assignments will culminate into the development of a fully functioning iOS application. As a final project, each student will design and implement an application of their choice to be presented in class. Each student will also be required to present a case study featuring an app from the Apple’s App Store. The studies will include a technical decomposition of the implementation (i.e. features, functionality, design, etc.) and a market analysis (i.e. competition, pricing, positioning, etc.) for the app. These case studies are designed to encourage students to gain an appreciation for the decisions companies and developers face when entering the app market.

**Machine Learning (MPCS 53111)**
This course introduces the fundamental concepts and techniques in data mining, machine learning, and statistical modeling, and the practical know-how to apply them to real-world data through Python-based software. The course examines in detail topics in both supervised and
unsupervised learning. These include linear and logistic regression and regularization; classification using decision trees, nearest neighbors, naive Bayes, boosting, random trees, and artificial neural networks; clustering using k-means, expectation-maximization, hierarchical approaches, and density-based techniques; and dimensionality reduction through PCA and SVD. Students use Python and Python libraries such as NumPy, SciPy, matplotlib, and pandas for implementing algorithms and analyzing data.

**Numerical Methods (MPCS 58001)**
This is a practical programming course focused on the basic theory and efficient implementation of a broad sampling of common numerical methods. Each topic will be introduced conceptually followed by detailed exercises focused on both prototyping (using matlab) and programming the key foundational algorithms efficiently on modern (serial) architectures. The ideal student in this course would have a strong interest in the use of computer modeling as predictive tool in a range of disciplines -- for example risk management, optimized engineering design, safety analysis, etc. The numerical methods studied in this course underlie the modeling and simulation of a huge range of physical and social phenomena, and are being put to increasing use to an increasing extent in industrial applications. After successfully completing this course, a student should have the necessary foundation to quickly gain expertise in any application-specific area of computer modeling. A familiarity with or strong interest in basic concepts of calculus and linear algebra will be helpful.

**OO Architecture: Patterns, Technologies, Implementations (MPCS 51050)**
This course gives hands-on experience in architecture and design and the communication of such designs in the form of patterns. There are no formal prerequisites except solid familiarity with Java and optionally familiarity with C++. The course is designed to give students a fundamental introduction to design and architectural patterns as they are implemented in large scale system architectures currently used in industry. Students will be encouraged to explore the various implementation possibilities afforded by these patterns. Trade-offs in terms of performance, development time, maintenance impact, etc. will also be discussed. Students will gain exposure to several industry-leading tools including Apache ActiveMQ and ServiceMix.

**Product Management (MPCS 51240)**
Product management is a cross-disciplinary endeavor that sits at the intersection of software engineering, marketing, and the user experience. Product managers are expected to create products in support of business objectives, ensuring that products deliver value to customers and are feasible to build within varying sets of constraints. In this course we will introduce the role of the product manager and demonstrate the challenges faced by product managers. We will explore approaches for managing the tension that exists between software development and product delivery using the minimum viable product and the product roadmap as critical tools. Key themes covered in this course include:

- Identifying a lack of product management
- Getting to market quickly; aka: “Perfection is the enemy of progress”
- Understanding the value of iteration and failing fast; aka: “You don’t know what customers value”
Delivering a product that meets customer needs; aka. “Customers don’t know what they need"
Trading off product focus and business needs; aka: “Drive on the freeway but know when to use the emergency lane”
Understanding technology maturity and product life cycles; aka: “You gotta know when to hold ‘em and know when to fold ‘em”

Coursework will include a mix of case studies, in-class workshops, product design sessions, written essays, and presentations. A key element of product management is communication, so there is an expectation that every student will contribute regularly to class discussions, and participation will be a substantial component of the final grade.

React Native App Development (MPCS 51034)
Cross platform development remains the holy grail of mobile development. While there have been many attempts to solve cross-platform development, all of the solutions to date have either added their own complex and idiosyncratic layer onto the software stack, or been woefully lacking in performance. React Native is different; it uses native iOS and Android components, and thus performance is at parity of native iOS and Android. Perhaps the killer application of React Native is OTA or Over The Air updates which allow developers to provide updates directly to users bypassing the App/Play stores. React Native is a game-changer for mobile technology. React is perhaps the first technology to realistically hold the promise of convergence among Android, iOS, and Web.

In this course we will cover the economics of cross-platform development and alternatives to React Native, the history of cross-platform and JavaScript, the foundations of JavaScript and then we will build several React Native applications during labs in class, covering Asynchronous, Networking, Navigation, Persistence and so on. By the end of this course, students will have a very good understanding of the cross-platform market, the key differentiators of React Native, and they will understand how to build React Native applications.

Time Series Analysis and Stochastic Processes (MPCS 58020)
Stochastic processes are driven by random events. They can be used to model phenomena in a broad range of disciplines, including science/engineering (e.g. computational physics, chemistry, and biology), business/finance (e.g. investment models and operations research), and computer systems (e.g. client/server workloads and resilience modeling). In many cases, relatively simple stochastic simulations can provide estimates for problems that are difficult or impossible to model with closed-form equations. In this class, we focus on the rudimentary ideas and techniques that underlie stochastic time series analysis, discrete events modeling, and Monte Carlo simulations. Course lectures will focus on the basic principles of probability theory, their efficient implementation on modern computers, and examples of their application to real world problems. Upon completion of the course, students should have an adequate background to quickly learn in depth specific Monte Carlo approaches in their chosen field of interest.

Trading Systems Design (MPCS 51026)
This class teaches the theory and practice of how to design a trading system.

Since 1998, after the U.S. Securities and Exchange Commission authorized electronic exchanges, the financial world has been using computer science extensively. The High Frequency Trading became a main actor of the main exchanges across the world.

During this course, we will learn how to create a reliable high-frequency trading system.

User Interface and User Experience Design (MPCS 51230)
This course is eleven weeks in length and covers the elemental practices of user interface (UI) design, user experience (UX) design, and user research. The intention of the course is to provide an overview of the experience design field so that the student is empowered to practice design as well as effectively manage design. It is, however, more likely that the student will collaborate with others on the design of products and services.

Experience design is the domain of empowering users to complete intended tasks efficiently and effectively. Research, UI, and UX help to create products and services that people want to use. User interface design (UI) is considered to include the visual design and organization of interfaces to guide users through the intended product experience. User experience design (UX) includes designing applications for users. It also includes the consideration of systems and business processes. Contemporary UX now includes topics such as service design, business design, API integration, and innovation. User research hopes to understand the needs and desires of customers. It includes topics such as voice of the customer research, customer engagement and retention measurement, as well as continual research to understand future product opportunities. Researchers also hope to understand if designs are usable by the intended audience.

During the class, we will utilize the latest design methods including prototyping, guerrilla research, design sprints, and Agile/Lean design/development processes.

At the conclusion of the class, the student will understand the value of using design processes in creating new products. The student will have researched a problem. They will have prototyped an online service and tested it with users, and have improved on their initial prototype afterwards. The student will have also created a sprint plan and prioritized the features for a minimal viable product. And finally, the students will present their solution to the class with data supporting their solution.

Web Development (MPCS 52553)
This course provides students with an introduction to modern web development, with an emphasis on the pragmatic skills needed to build live, functioning web applications. Students will learn fundamental domain modeling skills, HTML and CSS frameworks, agile software techniques and best practices, Javascript and AJAX, and both server-side and client-side debugging techniques. We will use the Ruby language and the Rails framework to immerse students into the challenge of building a live, database-backed web application deployed at a public web address. Specifically, students will learn how to: Build a live website or web
application and deploy it to the public internet; Use the Ruby on Rails framework to rapidly build a web application; Write software using the Ruby programming language; Use a relational database to provide content for dynamic websites; Follow industry best-practices of modern web software development; Troubleshoot and resolve the most common problems with web applications.