# CS Program Update

Proposal V4

Alexander(Sasha) Apartsin. 28/04/25

# Content

# 1. Trends and Motivation

Job market evolution and technological shifts

The evolution of CS/SE profession

The rise of AI-assisted software engineering

# 1.1 Software job market

*Top 5 trends before AI arrival*

**1. Software job roles are more specialized than ever**

Each role requires a distinct set of skills

**2. Employers expect new hires to deliver value from day one**

Immediate contribution is now the standard

**3. The market prioritizes technical expertise and proven experience**

Hands-on skills are considered just as important as diplomas and theory

**4. Technologies and markets are evolving at an unprecedented pace**

Existing technologies become outdated quickly

**5. Graduates will enter the era of AI-assisted software engineering**

Technical writing drives AI-assisted software construction

**Problem Solver**

- Tackles well-defined, often algorithmic problems by writing code from scratch.
- ***Key Skills:*** Strong grasp of core CS concepts and solid programming ability.
- Till 2000s

**Software Systems Builder**

- Develops large-scale software using foundational infrastructure like operating systems, databases, application servers, and standard libraries.
- ***Key Skills:*** System architecture, SDLC knowledge, OS, networking, and database expertise.
- Till 2010s

**Technology Stack Specialist**

- Expert in specific technologies or phases of development (e.g., full-stack, mobile, data, cloud, QA, architecting).
- ***Key Skills:*** Deep knowledge of tech stacks or lifecycle stages; adaptability to new tools and frameworks.
- Till 2020s

**AI-empowered Software Development Manager**

- Manage teams of specialized AI agents for designing and delivering complex software systems.
- ***Key Skills:*** Proficiency across multiple tech stacks and the full development lifecycle, product leadership, and effective technical communication.
- From 2025?

**We are here**

# 1.2 CS/SE Profession Evolution

Alexander(Sasha) Apartsion. 28/04/25

# 1.3 The Rise of AI-Assisted Software Engineering

*Emerging tasks, required skills, and skill gaps*

## Unambiguously specify requirements to AI

**Required skills**: strong technical writing and communication skills

**Gap**: coding skills alone are not enough

## Use AI for building systems involving multiple technologies

**Required skills**: broad understanding of diverse technology stacks

**Gap**: In-depth knowledge of a single technology is not enough

## Coordinate all phases of software development using AI

**Required skills**: proficiency across the entire Software Development Cycle: requirements engineering, testing, and deployment

**Gap**: Focusing on a single SDLC activity is not enough

# 2. Current CS Program

Current program structure

Current gaps

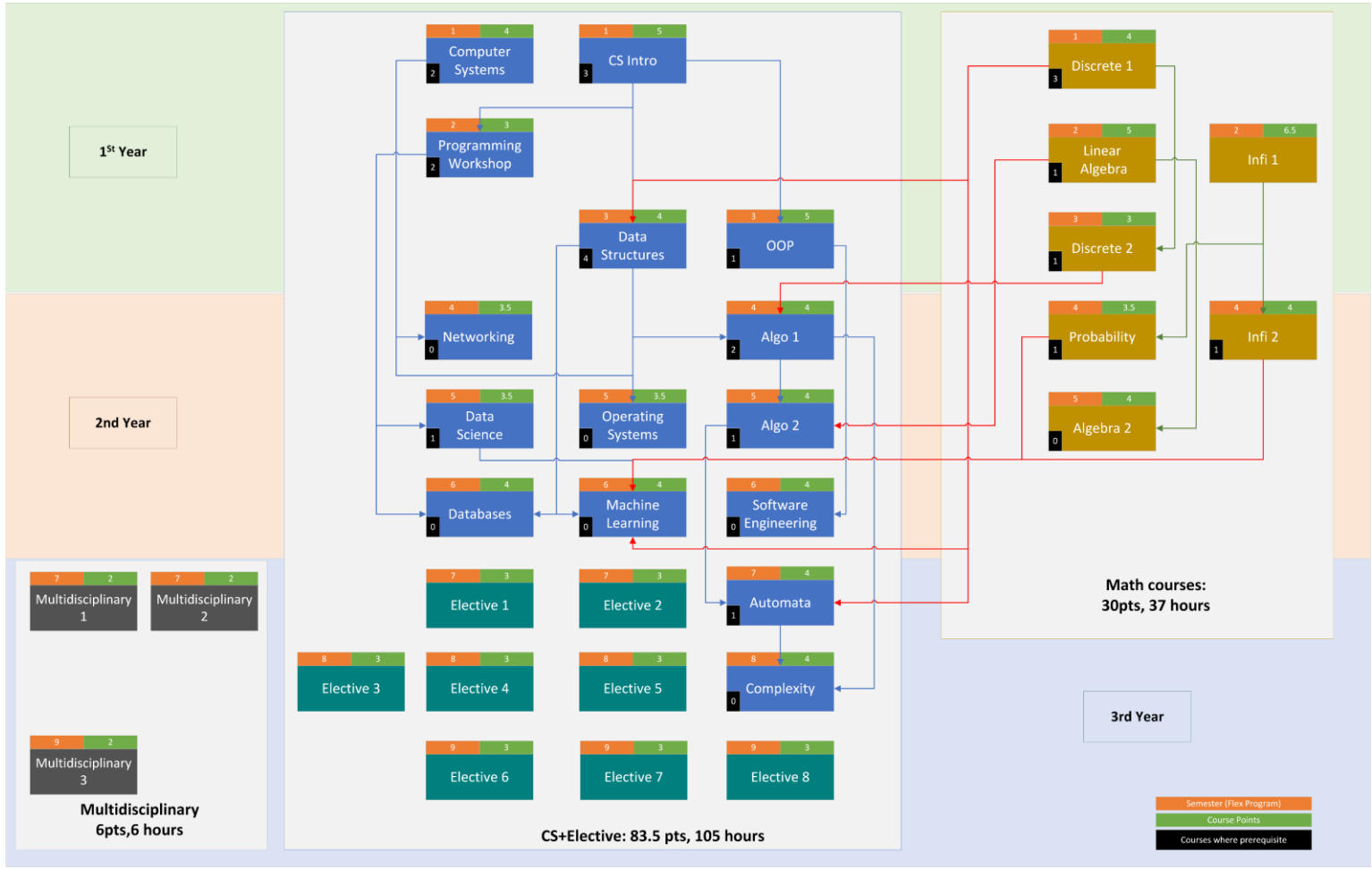
# 2.1 Current CS Program: Summary

## By course types

| מספר קורסים | סה"כ נ"ז | אחוז נ"ז | |
|---|---|---|---|
| 7 | 30 | 25% | חובה מתמטיים |
| 15 | 59.5 | 50% | חובה מדעי מחשב |
| 8 | 24 | 20% | בחירה מדעי מחשב |
| 3 | 6 | 5% | רב-תחומיים |
| **33** | **119.5** | **100%** | סה"כ |

## By year/semesters

| Year | Semester | Math | Mandatory | Elective | Multidisciplinary | Total |
|---|---|---|---|---|---|---|
| **1** | 1 | 4.0 | 9.0 | 0.0 | 0.0 | **13.0** |
| | 2 | 11.5 | 3.0 | 0.0 | 0.0 | **14.5** |
| | 3 | 3.0 | 9.0 | 0.0 | 0.0 | **12.0** |
| **2** | 4 | 7.5 | 7.5 | 0.0 | 0.0 | **15.0** |
| | 5 | 4.0 | 11.0 | 0.0 | 0.0 | **15.0** |
| | 6 | 0.0 | 12.0 | 0.0 | 0.0 | **12.0** |
| **3** | 7 | 0.0 | 4.0 | 6.0 | 2.0 | **12.0** |
| | 8 | 0.0 | 4.0 | 9.0 | 2.0 | **15.0** |
| | 9 | 0.0 | 0.0 | 9.0 | 2.0 | **11.0** |
| | | **30.0** | **59.5** | **24.0** | **6.0** | **119.5** |




[Complete list of mandatory courses]

Alexander(Sasha) Apartsion. 28/04/25

# 2.2 Current Program

Mandatory courses

Flex Track

| Year | Smstr | Pts | Course | Course Highlight* |
|---|---|---|---|---|
| 1st year 39.5pts 9 courses | 1 13pts | 5 | Introduction to Computer Science | C, rudimentary algo |
| | | 4 | Computer Systems | Boolean ops, assembly, architecture/ALU, gates |
| | | 4 | Discrete Mathematics I | Logic, Number Theory, Functions, Induction, Relation, Sets |
| | 2 14.5pts | 3 | Programming Workshop | Data Structures in C, advanced C |
| | | 5 | Linear Algebra 1 | Analytic geometry, system of linear eqs, matrices, linear spaces |
| | | 6.5 | Calculus 1 | Function study, asymptotes,, Integrals, Limits |
| | 3 12pts | 4 | Data Structures | Recurrence, Big-O, Trees, Set-Union, Hash, Sorting |
| | | 5 | Object Oriented Programming | C++, Classes/Objects/Inheritance |
| | | 3 | Discrete Mathematics 2 | Combinatorics, Modulo Arithmetic |
| 2nd year 42pts 11 courses | 4 15pts | 3.5 | Networking | E2E Networking Layers |
| | | 4 | Algorithms 1 | DFS/BFS, Complexity, Search, Strings |
| | | 3.5 | Probability | RV, expectation. PDF |
| | | 4 | Calculus 2 | Gradient, Taylor, Multivariate Function Study, Integrals |
| | 5 15pts | 3.5 | Data Science | Pandas/NumPy,, Visualization, PCA, Regression/Classification, Metrics |
| | | 3.5 | Operating Systems | Spooling, Concurrency, Virtual Memory, Interrupts, Processes |
| | | 4 | Algorithms 2 | Graph algorithms, matching, flow, traversal |
| | | 4 | Algebra 2 | Inner product, groups, SVD, eigenvalues |
| | 6 12pts | 4 | Databases | SQL, Relation Algebra, Normalization, ERD |
| | | 4 | Machine Learning | Regression, Neural Networks, Classification, SVM, KNN, Feature Selection |
| | | 4 | Software Engineering | SCRUM, Design Patterns, UML, Testing |
| 3rd year 8pts 2 courses | 7(4pts) | 4 | Automata | Regular Expression, Context-Free, Formal Languages |
| | 8(4pts) | 4 | Complexity | Turning Machines, Complexity Classes |

# 2.3 Current Curriculum: Gaps

1. Heavy emphasis on abstract mathematics and **theoretical** foundations

2. **Disconnected** sequencing and **overlap** across core and elective courses

3. **Delayed exposure to practical**, industry-relevant skills

4. Insufficient opportunities for hands-on learning and **demonstrable project work**

5. **Outdated technological toolset** and practices

6. Lack of **specialization toward defined job roles** or application domains

# 3. Proposed Update

Objectives and target skills map(skills students should have by 1st/2nd/3rd year)

Mandatory courses and concentration cores

Benefits and encounters with AI

# 3.1 Curriculum Update: Objectives

## 1. Project-based learning

- 1.1 Ramp up in-course hands-on assignments and projects
- 1.2 Introduce project-based workshop elective courses
- 1.3 Create H.I.T. hosted students' course projects and assignment page

## 2. Concentrate electives

- 2.1 Introduce 2-3 concentrations based on specific job roles
- 2.2 Allocate resources(staff, tools) to concentration core courses
- 2.3 Optimize resources by cutting low-demand and redundant elective courses

## 3. Compress time-to-value

- 3.1 Introduce practical material starting from the first year
- 3.2 Drop some mandatory courses
- 3.3 Consolidate certain mandatory courses

## 4. Align curriculum

- 4.1 Simplify course dependencies and ensure logical progression between courses
- 4.2 Align course syllabi to eliminate content gaps and overlaps.
- 4.3 Update course programming language and modern technological stack

## 5. Monitor Feedback

- 5.1 Perform more student surveys
- 5.2 Monitor course grade distribution
- 5.3 Introduce an internal/external review process

# 3.2 Target Skills Map

- ***Not a course list***.
- *Skills we'd like students to have by the end of each year*

[Concentration Core Electives]

| Year | Target Skills | Subjects | Concepts (illustration) | Tools/Languages |
|---|---|---|---|---|
| 1st | Understand **basic math for CS**. | Discrete Math | Relations, Combinatorics | Python and visualization, where possible (see textbook) |
| | | Linear Algebra | Vectors, Linear Spaces, Dot-Product | |
| | | Calculus | Functions, Derivatives, Optimization | |
| | | Probability and Statistics | RV, distributions, expectations | |
| | Understand how **computers work**. | Hardware | Gates/ALU/Memory/Assembly | HDL/ASM |
| | | Software | Operating systems/Compilers and Languages | Windows, Python/C++ |
| | Understand and use **basic algorithms** | Basic data structures | List/Hash/Trees | Python |
| | | Basic algorithms | Search and sorting, DFS/BFS | Python |
| | | Basic complexity Analysis | Big-O and asymptotic complexity | **N/A** |
| | Build **simple software systems** | Programming languages | Control structures, variables, functions | C/Python |
| | | DB/Web | Markup, Relation DB, Web App | HTML/CSS, SQL |
| | | AI Services and Tools | Cloud API, AI-enabled IDE | OpenAI API, GitHub Co-pilot |
| 2nd | Understand and use **advanced algorithms** | Advanced Algorithms | Randomized/online, computational geometry | Python |
| | | Computational Models | Automata, Turning ,complexity classes | **N/A** |
| | Build and use **basic AI/ML models** | Data Engineering | Data storage and processing, cleaning | Python/Pandas/Matplot |
| | | Statistical Inference | Estimators, Hypothesis Testing | Python |
| | | Machine Learning | Classical regression/classification, Intro to NN | Python, Scikit, PyTorch |
| | Build **complex software systems** using modern methodologies and tools | Object-Oriented Software | OOP/OOD, Classes and Objects | Java/C++ |
| | | Software Engineering Process | SDLC: Software Development Lifecycle | Git, Jira |
| | | Software architecting and design | Client-Server/n-Tier, GoF design patterns | Java/C++ |
| | | Basic OS/Network/Cloud services | Process/Thread+ TCP/IP/HTTP+APIs | C++/Python/Sockets/OpenAI |
| 3rd | Acquire skills and experience in **selected concentration** | **Track 1**: AI Engineering | Deep Learning, GenAI, NLP, Vision, Big Data | PyTorch,/Transformers |
| | | **Track 2**: Software Engineering | Web, Databases, AI-assisted SE, DevOps, Mobile | JS/Cloud/Redis/Kafka |
| | | **Track 3**: Cybersecurity/Networking | App/Cloud/HW Cybersecurity, AI for cyber | C/Wireshark/Python |

# 3.3 New Program: Mandatory courses 80 pts, 14 courses(Flex)

| Year | Smstr | Pts | Course+ Example of textbooks and existing courses | Syllabus Highlights | Coding Practice |
|---|---|---|---|---|---|
| 1st year 7 Courses 40 pts | 1 12pts | 6 | **Discrete Mathematics and Probability** Lectures, Recitations Based on: MIT 6.042J Mathematics for CS | Induction, Numbers, Graphs, Relations, Recurrence, Combinatorics, Probability | Python graph/probability **assignments** |
| | | 6 | **Introduction to Computer Science and AI** Lectures Based on: Harvard's CS50, Introduction to CS, C50.AI | C, Arrays, Algorithms, Memory, Data Structures, Python, SQL, HTML, Flask, **AI** | Python/HTML/SQL **project** |
| | 2 16pts | 5 | **Single-Variable Calculus** Video 1, Based on: Math 100, Univ. of Cincinnati | Functions, Limits, Continuity, Derivatives, Integration | Python symbolic derivatives **assignment** |
| | | 6 | **Computer Systems** Syllabus, P1 Video, P2 Video Based on: HUJI's Nand2Tetris | Boolean logic, gates, memory, and assembler, VM, Compilers, Operating Systems | HDL/Python **project** |
| | | 5 | **Linear Algebra** Lectures, Code Based on: Coding The Matrix, Brown University | Complex numbers, fields, vectors, inner product, vector space, linear systems, change of basis, eigenvectors, SVD | Python numerical algebra **assignments** |
| | 3 12pts | 6 | **Algorithms and Data Structure** Lectures, Video Based on: MIT's 6.006 Introduction to Algorithms | Sorting, Hashing, Trees, DFS/BFS, Dynamic Programming, Complexity Basics | Python algo **assignments** |
| | | 6 | **Multivariate Calculus and Optimization** Video Based on: Math200 , University of Victoria | Inner product, Geometry, Gradient, Jacobian, Optimization | Python optimization **assignments** |
| 2nd year 7 Courses 40 pts | 4 16pts | 6 | **Software Design and Architecting** Alberta/Coursera | Classes/Objects/Inheritance/OOP/OOD | Java/C++ **project** |
| | | 5 | **Advanced Algorithms** | Combinatorial Optimization, Number Theory, Randomized Algorithms | Python algo **assignments** |
| | | 5 | **Models of Computations** MIT 18404J | Automata + Turning+ Complexity | **NA** |
| | 5 12pts | 6 | **Operating Systems and Networking** OpenCSF, CS341 | Process/Thread/VMem/IPC/Network/APIs | C/Python/Linux/Sockets **project** |
| | | 6 | **Data Engineering and Analysis ,** dsc100+dsc10 | (NO)SQL, Data Imputations/Cleaning, Visualization, Basic DS | SQL/Pandas/NumPy **project** |
| | 6 12pts | 6 | **Statistical Inferencing and Machine Learning** | Statistics, Classification, Regression, **GenAI basics** | Python/scikit-learn **project** |
| | | 6 | **Software Engineering Methodologies and Tools** | Software Development Lifecycle, **AI SE tools** | SDLC/QA/Git/Jira **project** |

# 3.7 New Program: Mandatory courses 80 pts, 14 courses(Regular)

| Year | Smstr | Pts | Course | Syllabus Highlights | Coding Practice |
|---|---|---|---|---|---|
| 1st year 7 Courses 40 pts | 1 17pts | 6 | **Discrete Mathematics and Probability** Lectures, Recitations Based on: MIT 6.042J Mathematics for CS | Induction, Numbers, Graphs, Relations, Recurrence, Combinatorics, Probability | Python graph/probability **assignments** |
| | | 6 | **Introduction to Computer Science and AI** Lectures Based on: Harvard's CS50, Introduction to CS CS50.AI | C, Arrays, Algorithms, Memory, Data Structures, Python, SQL, HTML, Flask, **AI** | Python/HTML/SQL **project** |
| | | 5 | **Single-Variable Calculus** Video 1, Based on: Math 100, Univ. of Cincinnati | Functions, Limits, Continuity, Derivatives, Integration | Python symbolic derivatives **assignment** |
| | 2 23pts | 6 | **Algorithms and Data Structure** Lectures, Video Based on: MIT's 6.006 Introduction to Algorithms | Sorting, Hashing, Trees, DFS/BFS, Dynamic Programming, Complexity Basics | Python algo **assignments** |
| | | 6 | **Computer Systems** Syllabus, P1 Video, P2 Video Based on: HUJI's Nand2Tetris | Boolean logic, gates, memory, and assembler, VM, Compilers, Operating Systems | HDL/Python **project** |
| | | 5 | **Linear Algebra** Lectures, Code Based on: Coding The Matrix, Brown University | Complex numbers, fields, vectors, inner product, vector space, linear systems, change of basis, eigenvectors, SVD | Python numerical algebra **assignments** |
| | | 6 | **Multivariate Calculus and Optimization** Video Based on: Math200 , University of Victoria | Inner product, Geometry, Gradient, Jacobian, Optimization | Python optimization **assignments** |
| 2nd year 7 Courses 40 pts | 3 22pts | 6 | **Software Design and Architecting,** Alberta/Coursera | Classes/Objects/Inheritance/OOP/OOD + Patterns | Java/C++ **project** |
| | | 5 | **Advanced Algorithms** | Combinatorial Optimization, Number Theory, Randomized Algorithms | Python algo **assignments** |
| | | 5 | **Models of Computations** MIT 18404J | Automata + Turning+ Complexity | **NA** |
| | | 6 | **Data Engineering and Analysis** dsc100+dsc10 | (NO)SQL, Data Imputations/Cleaning, Visualization, Basic DS | SQL/Pandas/NumPy **project** |
| | 4 18pts | 6 | **Operating Systems and Networking** OpenCSF, CS341 | Process/Thread/VMem/IPC/Network/APIs | C/Python/Linux/Sockets **project** |
| | | 6 | **Statistical Inference and Machine Learning** | Statistics, Classification, Regression, **GenAI basics** | Python/scikit-learn **project** |
| | | 6 | **Software Engineering Methodologies and Tools** | Software Development Lifecycle, **AI SE tools** | SDLC/QA/Git/Jira **project** |

# 3.4 Concentrations core courses

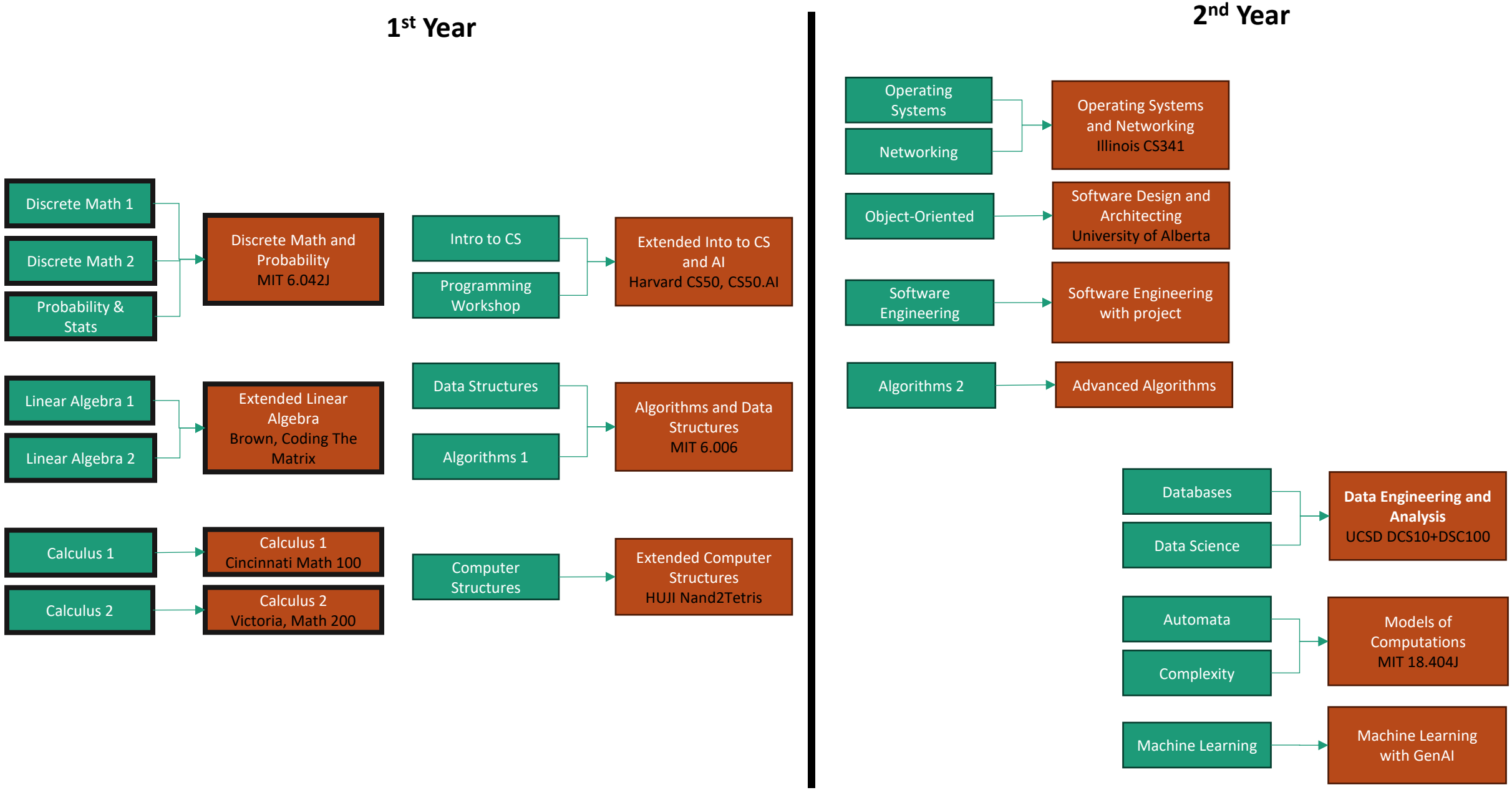
*Additional high-quality elective courses are desirable*

| | |
|---|---|
| **Artificial Intelligence**<br><br>Top Roles<br>AI/ML Engineer<br>Data Scientist | 1. **Deep Learning 60311**<br>**2. Deep Models for GenAI( Sasha)**<br>**3. Big Data Algorithms**<br>4. **Reinforcement Learning and Multiagent Systems**<br>5. **Computer Vision 65212**<br>**6. Natural Language Processing 65339(Sasha)**<br>**7. Robotics and Autonomous Systems 69983**<br>8. **AI Models for Software Engineering(Sasha)** |
| **Software Engineering**<br><br>Top Roles<br>Full-Stack Engineer<br>Application Developer | 1. **Cloud Computing(Andrey)**<br>2. **Design of AI-based and Data Intensive Systems**<br>3. **Secure and Correct Software Development 64444**<br>4. **Web Development 1: Backend 65356**<br>5. **Web Development 2: Front-End 65363**<br>6. **DevOps Principles 65353**<br>**7. Mobile and IoT Development 65336**<br>8. **AI-Assisted Software Construction** |
| **Cybersecurity**<br><br>Top Roles<br>Cybersecurity Analyst<br>Security Engineer | 1. **Cryptography 65330**<br>2. **Application Security 65337**<br>3. **Network Security 65338**<br>4. **Hardware Security**<br>5. **Cloud Security**<br>6. **Security and Risk Governance**<br>7. **Blockchain Technologies 67007**<br>8. **AI for Cybersecurity(Andery)** |

***In preparation***

***Gaps ,need to be developed***

***Exist, might require update***




# Current2New Course Map

# 3.5 Prerequisites

| Smstr | Course | Prerequisite Courses | Opens Courses |
|---|---|---|---|
| **1** | **Discrete Mathematics and Probability** | None | Single-Variable Calculus(**2**)<br>Linear Algebra(**2**) |
| **1** | **Introduction to Computer Science and AI** | None | Computer Systems(**2**)<br>Algorithms and Data Structures(**3**) |
| **2** | **Single-Variable Calculus** | Discrete Mathematics and Probability(**1**) | Multivariate Calculus and Optimization(**3**) |
| **2** | **Computer Systems** | Introduction to Computer Science(**1**) | Software Design and Architecting(**4**) |
| **2** | **Linear Algebra** | Discrete Mathematics and Probability(**1**) | Algorithms and Data Structures(**3**)<br>Multivariate Calculus and Optimization(**3**) |
| **3** | **Algorithms and Data Structures** | Introduction to Computer Science(**1**)<br>Linear Algebra(**2 or ||**) | Advanced Algorithms(**4**)<br>Models of Computations (**4**)<br>Operating Systems and Networking (**5**)<br>Data Engineering and Analysis (**5**) |
| **3** | **Multivariate Calculus and Optimization** | Single-Variable Calculus(**2**)<br>Linear Algebra(**2 or ||**) | Statistical Inference and Machine Learning(**6**)<br>**Graduate Degree** |
| **4** | **Software Design and Architecting** | Computer Systems(**2**) | Operating Systems and Networking(**5**)<br>Data Engineering and Analysis (**5**)<br>Software Engineering Methodologies and Tools(**6**) |
| **4** | **Advanced Algorithms** | Algorithms and Data Structure(**3**) | **Cyber Concentration** |
| **4** | **Models of Computations** | Algorithms and Data Structure(**3**) | **Graduate Degree** |
| **5** | **Operating Systems and Networking** | Algorithms and Data Structure(**3**)<br>Software Design and Architecting(**4**) | Software Engineering Methodologies and Tools(**6**) |
| **5** | **Data Engineering and Analysis** | Algorithms and Data Structure(**3**)<br>Software Design and Architecting(**4 or ||**) | Statistical Inferencing and Machine Learning(**6**) |
| **6** | **Statistical Inferencing and Machine Learning** | Multivariate Calculus and Optimization(**3**)<br>Data Engineering and Analysis (**5**) | **AI Concentration** |
| **6** | **Software Engineering Methodologies and Tools** | Software Design and Architecting(**4**)<br>Operating Systems and Networking(**5 or ||**) | **SW Concentration** |

# 3.6 Encounters with DataScience, AI and Machine Learning in Mandatory Courses

| Smstr | Course | Example of illustrations /use cases |
|---|---|---|
| 1 | **Discrete Mathematics and Probability** | Probabilistic Bayesian reasoning with Python, Naïve Bayes |
| 1 | **Introduction to Computer Science and AI** | AI /Projects in CS50 CS50AI, CS50, Python |
| 2 | **Single-Variable Calculus** | Derivative for minima/model sensitivity, Taylor expansion for approximation/LIME, Python /SymPy |
| 2 | **Algorithms and Data Structure** | Bloom Filters for spam, kNN search with KD-Trees, A* planning, Python exercises |
| 3 | **Linear Algebra** | Dimensionality reduction/PCA, ML regression in vector form and closed form solution, Python |
| 3 | **Multivariate Calculus and Optimization** | Gradient descent optimization for ML and neural networks, Python |
| 4 | **Software Design and Architecting** | AI code generation |
| 5 | **Data Engineering and Analysis** | Python Pandas/NumPy/Databases |
| 5 | **Operating Systems and Networking** | Network/REST : Cloud AI APIs(OpenA) and their use |
| 6 | **Statistical Inference and Machine Learning** | ML Models |
| 6 | **Software Engineering Methodologies and Tools** | AI tools for software engineering |

# 4. Concentrations Core Courses

Core Courses for AI/Software Engineering and Cyber Concentrations

Alexander(Sasha) Apartsion. 28/04/25

# 4.1 AI/ML concentration: Core Courses

| # | Course No | Course | Concepts | Practice |
|---|---|---|---|---|
| 1 | **60311** | Deep Learning | Classical DL: CNN/RNN+ optimization | PyTorch, TensorBoard |
| 2 | **Sasha** | Deep Learning Models for GenAI | GenAI DL: Diffusion/Transformers/GAN/VAE | PyTorch, HuggingFace |
| 3 | | Big Data Algorithms | Map-Reduce Algorithms, LSH, Streaming | PySpark, Ray |
| 4 | | Reinforcement Learning and Multiagent Systems | MDP, MAB, Q/Value leaning, Deep RL | OpenAI Gym, SB3, RLlib |
| 5 | **65212** | Computer Vision | DL/GenAI for classification/segmentation/detection | OpenCV, Ultralytics |
| 6 | **65339** | Robotics and Autonomous Systems | Kinematics/planning/navigation/sensors | ROS, libraries |
| 7 | **69983** | Natural Language Processing | DL/LLM for classification/generation/extraction | Spacy, NLTK, HuggingFace |
| 8 | **Sasha** | AI models for Software Engineering | Models for code generation/testing/documenting | HuggingFace, OpenHands |

# 4.2 SE/FS Concentration: Core Courses

| # | Course No | Course | Concepts | Practice |
|---|---|---|---|---|
| 1 | | Cloud Computing | IaaS/PaaS, Storage, Compute, Networks | AWS/Azure , Kubernetes |
| 2 | | Design of AI-based and data intensive systems | Scalable Architectures, Big Data ETL | Redis, Spark, Kafka, MLFLow |
| 3 | 64444 | Secure and Correct Software Development | Secure coding and QA | Selenium/Junit, TBD |
| 4 | 65363 | Web development: front-end | DOM, Responsive Design, Event-driven Architectures | CSS/React/TypeScript |
| 5 | 65356 | Web-development back-end | REST, Microservices, Layered Architecture | ASP.Net/Node, MongoDB |
| 6 | 65353 | DevOps Principles | CI/CD pipelines, observability | Jenkis, Docker, TerraForm, GitHub Actions |
| 7 | 65336 | Mobile and IoT development | Sensors, Actuators, Communications | Android, Arduino, MQTT |
| 8 | | AI-assisted software construction | AI-assisted Software Development Lifecycle | GitHub Co-Pilot, Cursor, Cline |

# 4.3 **Cyber Concentration**: Core Courses

| # | Course No | Course | Concepts | Practice(???) |
|---|---|---|---|---|
| **1** | **65330** | Cryptography | PKI algorithms, Homomorphic Encryption | OpenSSL, PyCryptodome |
| **2** | **65337** | Application Security | Secure coding, SSDLC, OWASP top 10 | OWASP ZAP |
| **3** | **65338** | Network Security | Intrusion Detection, Packet Inspection, TLS | Wireshark, PyShark |
| **4** | | Hardware Security | TPM, Side-Channel Attacks | ChipWhisperer |
| **5** | | Cloud Security | Identify and Access Management, Isolation | AWS boto3, Terraform |
| **6** | | Security and Risk Governance | Risk Management Lifecycle, Policies and Compliance, Risk Assessment | TBD |
| **7** | **67007** | Blockchain Technologies | Distributed Ledger, Smart Contracts | Web3.py |
| **8** | | AI for Cybersecurity | Anomaly Detection, Malware Classification, User Behavior Analysis | Scikit PyTorch |

# 5. Benefits and Challenges

Small number of extended mandatories with cross-functional content

Practice-driven and functional knowledge-driven teaching

Job role-based concentration tracks as project portfolio building process

# 5.1 Benefits and challenges

**Concentrated mandatory courses**

(math and basic CS)

- A smaller number of larger courses is easy to manage and align: 22 vs. 14 mandatory courses.
- Sourced from the existing textbooks/courses and well-known authors/universities.
- Each course has more F2F teaching hours (60-110 hours) compared to the source courses (30-50 hours)
- Result in a fast track to practical skills, enabling early industry internship (after the 2nd year and earlier)
- **Challenge**: bigger and project-based courses **require larger teaching teams**: lecturers, TAs, HW graders.

**Practice-driven teaching approach**

(where possible and makes sense)

- Prioritized conceptual and functional understanding (what it does vs. proofs or detailed factual knowledge).
- Application-oriented teaching: emphasizes using concepts in real-world and problem-solving contexts
- Project-based/hands-on focus: projects/coding assignments  almost everywhere
- Host all project/coding HW assignments at a single student repository/page for the project portfolio
- **Challenge**: Night requires a **multi-disciplinary teaching staff** (Math + CS TA) to introduce CS/Coding.

**Specialization via concentration core electives**

(with additional optional electives)

- Aligned elective core courses program based on the job-role skillset requirements
- Relies on course dependency, logical progression, and no content gaps/overlaps
- Students gradually build their project portfolio from mandatory to specialization.
- Concentrate resources for core courses (teams, AI APIs, cloud compute)
- **Challenge**: Need to continuously monitor for **relevancy/quality** , update courses' content if required

# 6. Implementation Strategy

Implementation options and action items

# 6.1 Implementation Options

- Option: I: Gradual rollout
  - Gradually replace/upgrade some courses while keeping others
  - **Challenges**:  content overlap and gaps

- Option II: Program Cutover
  - Start the new program from the following year
  - **Challenges**: system shock/havoc, students repeating courses

- **Option III: Parallel run**
  - Introduce “new track” 1st year courses and keep the old program
  - Start from a few concentrations as recommended elective course sequnce
  - Run a new program for the selected population: excellent students, IDF tech unit
  - Gradually reduce and phase out the old program after a few years

# 6.2 Parallel Run

- Introduce “extended” 1st year courses
  - 3 Core CS Courses (**extended** track)
    - **Extended** Introduction to Computer Science
    - **Extended** Algorithms and Data Structures
    - **Extended** Computer Systems
  - 4 Math Courses (for **computer science**)
    - Discrete Mathematics with Probability for **Computer Science**
    - Calculus I for **Computer Science**
    - Calculus II for **Computer Science**
    - Linear Algebra for **Computer Science**

- Pilot run
  - Keep old 1st year courses
  - Register for the “extended track” according to the selected criteria and demand
  - Enable the 2nd-year CS courses to be taken based on an extended version
  - Gradually roll out the second-year extended courses as well

- Start from a 1-2 specialization track
  - Revise  and align existing course syllabi, introduce new

# 6.3 Implementation action items

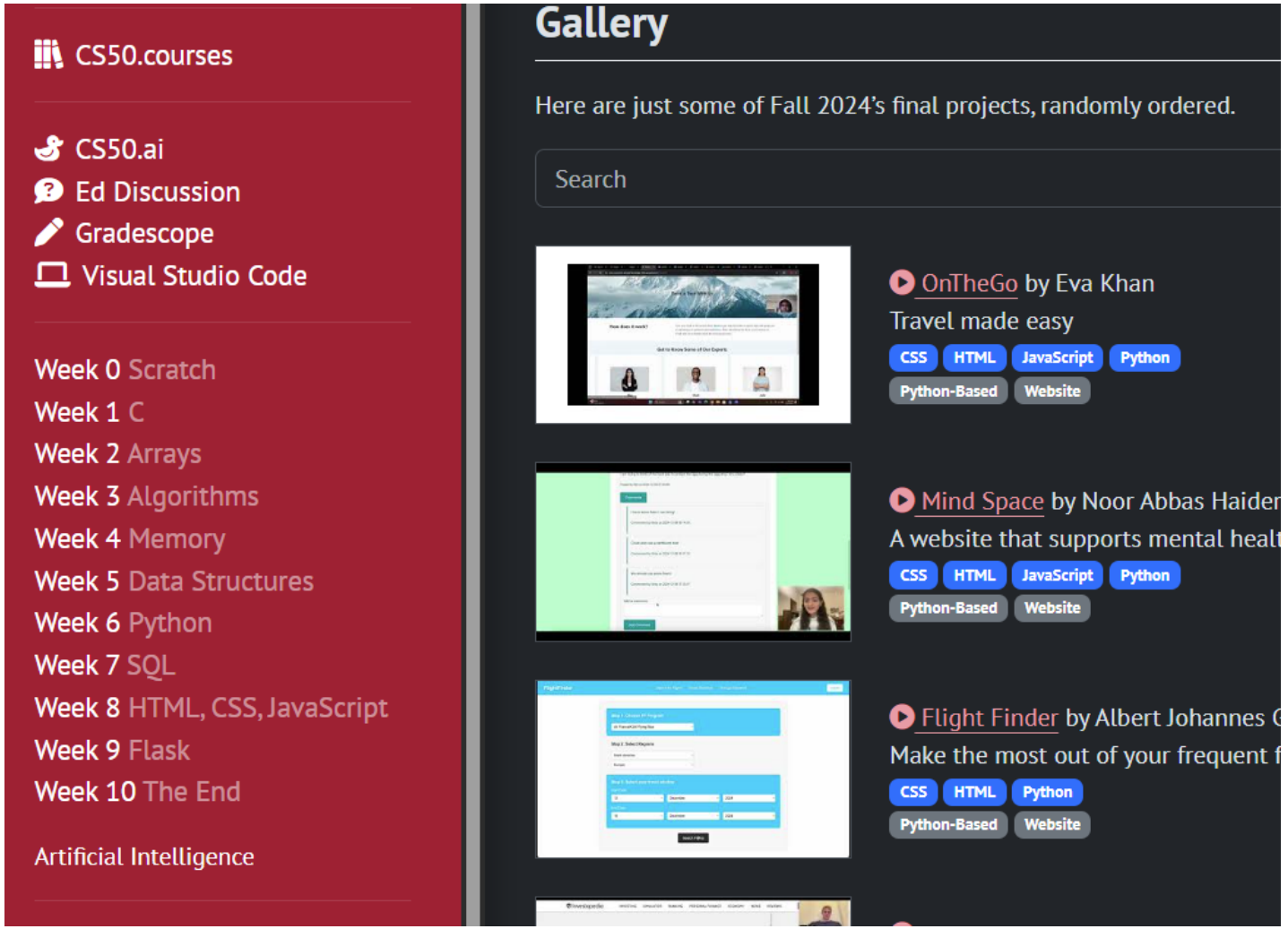
- Establish a board for each mandatory course
  - A dedicated leading team for each mandatory course
  - Cross-functional teams: math  and CS
  - Increase the supporting team, hire CS PhD candidates as Teaching Assistants

- Concentration courses
  - Design new courses, call for lecturers on missing courses
  - Review and extend existing courses
  - Verify with the industry (mid-level, hiring managers in the industry)

- Concentrate resources
  - Drop trivial, non-academic, and low-demand courses
  - Increase the number of teams on mandatory and core concentration courses
  - Invest in resources: AI APIs, cloud computing, etc.

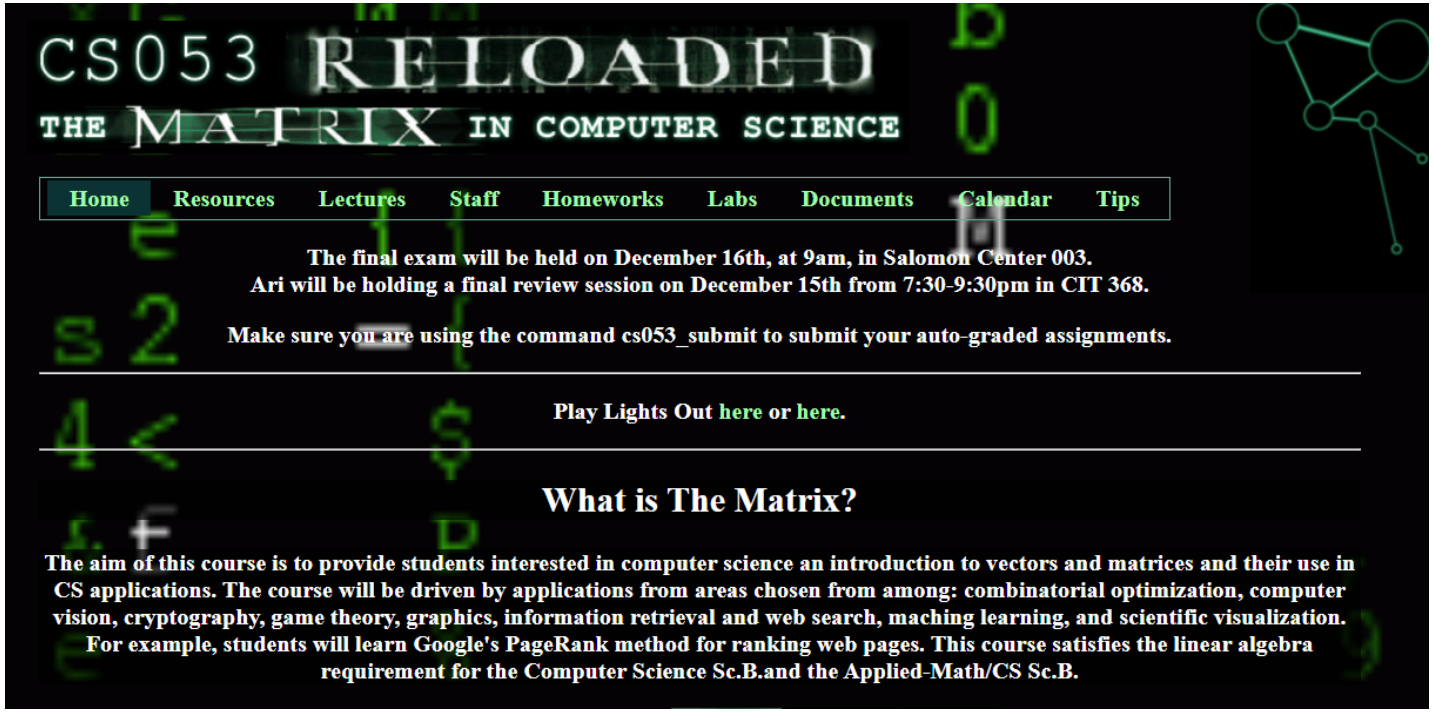
# 7. Helpful Resources

# Harvard CS50




Alexander(Sasha) Apartsion. 28/04/25

# Brown CS053

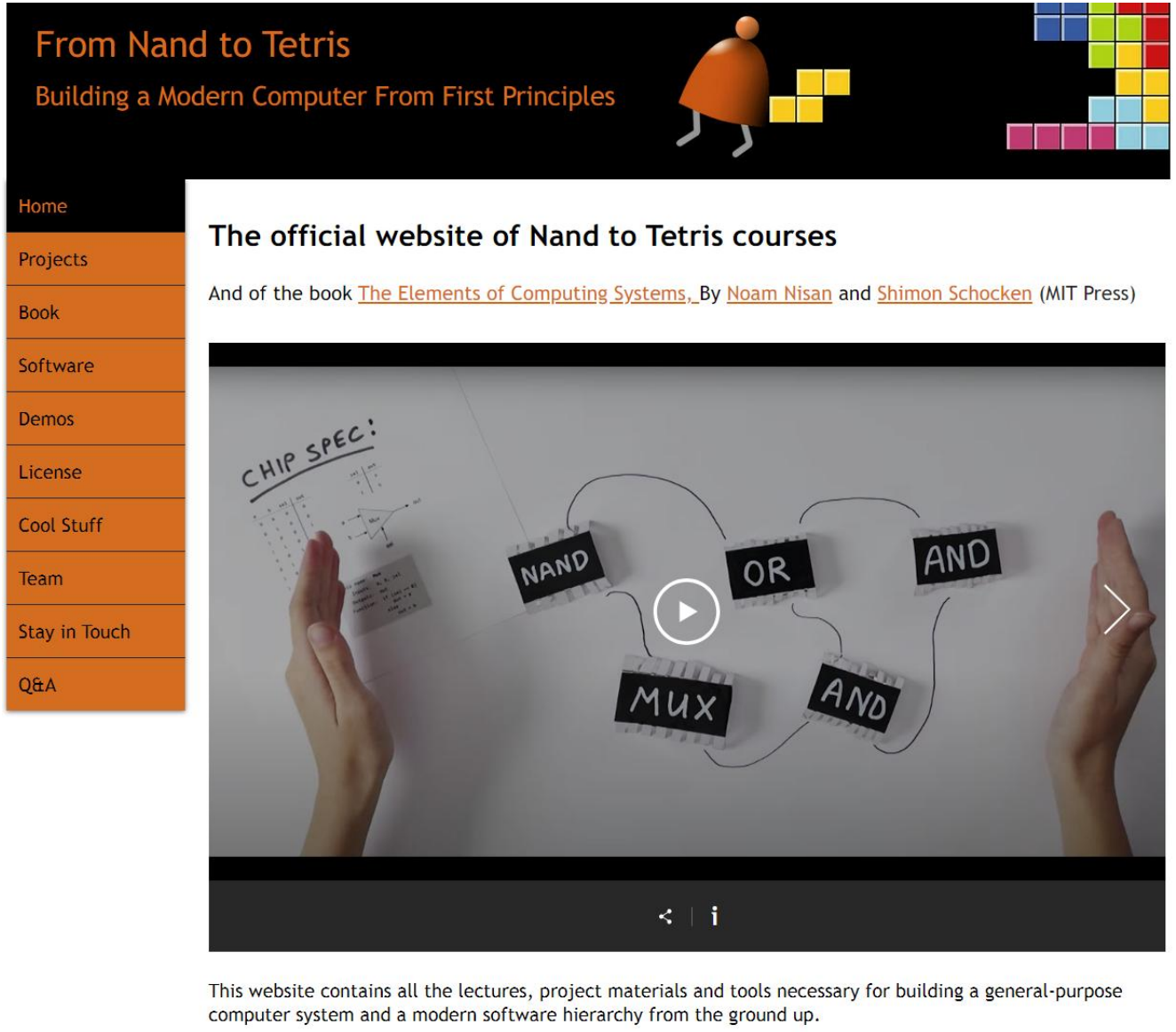

CS053 RELOADED
THE MATRIX IN COMPUTER SCIENCE
Home
Resources
Lectures
Staff
Homeworks
Labs
Documents
Calendar
Tips
The final exam will be held on December 16th, at 9am, in Salomon Center 003.
Ari will be holding a final review session on December 15th from 7:30-9:30pm in CIT 368.
Make sure you are using the command cs053_submit to submit your auto-graded assignments.
Play Lights Out here or here.
What is The Matrix?
The aim of this course is to provide students interested in computer science an introduction to vectors and matrices and their use in CS applications. The course will be driven by applications from areas chosen from among: combinatorial optimization, computer vision, cryptography, game theory, graphics, information retrieval and web search, maching learning, and scientific visualization. For example, students will learn Google's PageRank method for ranking web pages. This course satisfies the linear algebra requirement for the Computer Science Sc.B.and the Applied-Math/CS Sc.B.

# MIT 6.045J: (Discrete) Math for CS

| LEC # | TOPICS | KEY DATES |
|---|---|---|
| 1 | Introduction and proofs | |
| 2 | Induction | Problem set 1 due |
| 3 | Strong induction | |
| 4 | Number theory I | Problem set 2 due |
| 5 | Number theory II | |
| 6 | Graph theory and coloring | Problem set 3 due |
| 7 | Matching problems | |
| 8 | Graph theory II: minimum spanning trees | Problem set 4 due |
| 9 | Communication networks | |
| 10 | Graph theory III | Problem set 5 due |
| 11 | Relations, partial orders, and scheduling | |
| 12 | Sums | Problem set 6 due |
| 13 | Sums and asymptotics | |
| 14 | Divide and conquer recurrences | Problem set 7 due |
| | Midterm | |
| 15 | Linear recurrences | |
| 16 | Counting rules I | Problem set 8 due |
| 17 | Counting rules II | |
| 18 | Probability introduction | Problem set 9 due |
| 19 | Conditional probability | Problem set 10 due |
| 20 | Independence | |
| 21 | Random variables | Problem set 11 due |
| 22 | Expectation I | |
| 23 | Expectation II | Problem set 12 due |
| 24 | Large deviations | |
| 25 | Random walks | |

Alexander(Sasha) Apartsion. 28/04/25

# HUJI Nand2Tetris

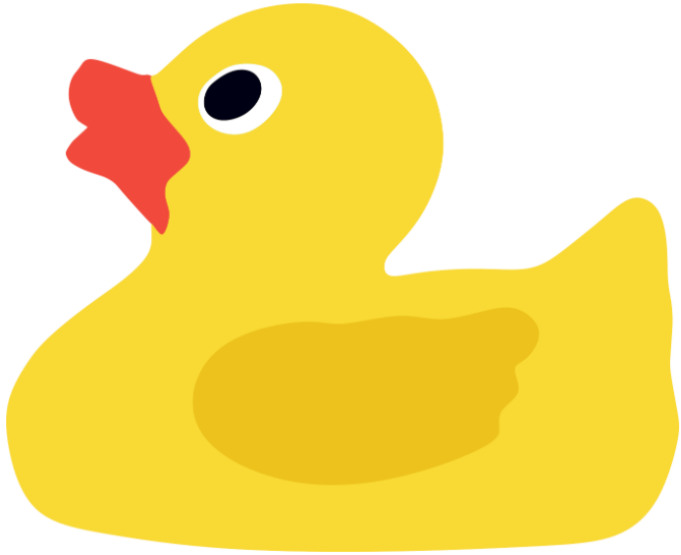


Illinois CS341 System programming (OS + Networking)





# Home



This coursebook is being built by students and faculty from the University of Illinois. It is based on a crowd-source authoring wikibook experiment by Lawrence Angrave from CS @ Illinois, but is now its own .tex based project. Its source code is located at the Github link which you can find a pdf version of the book as well.

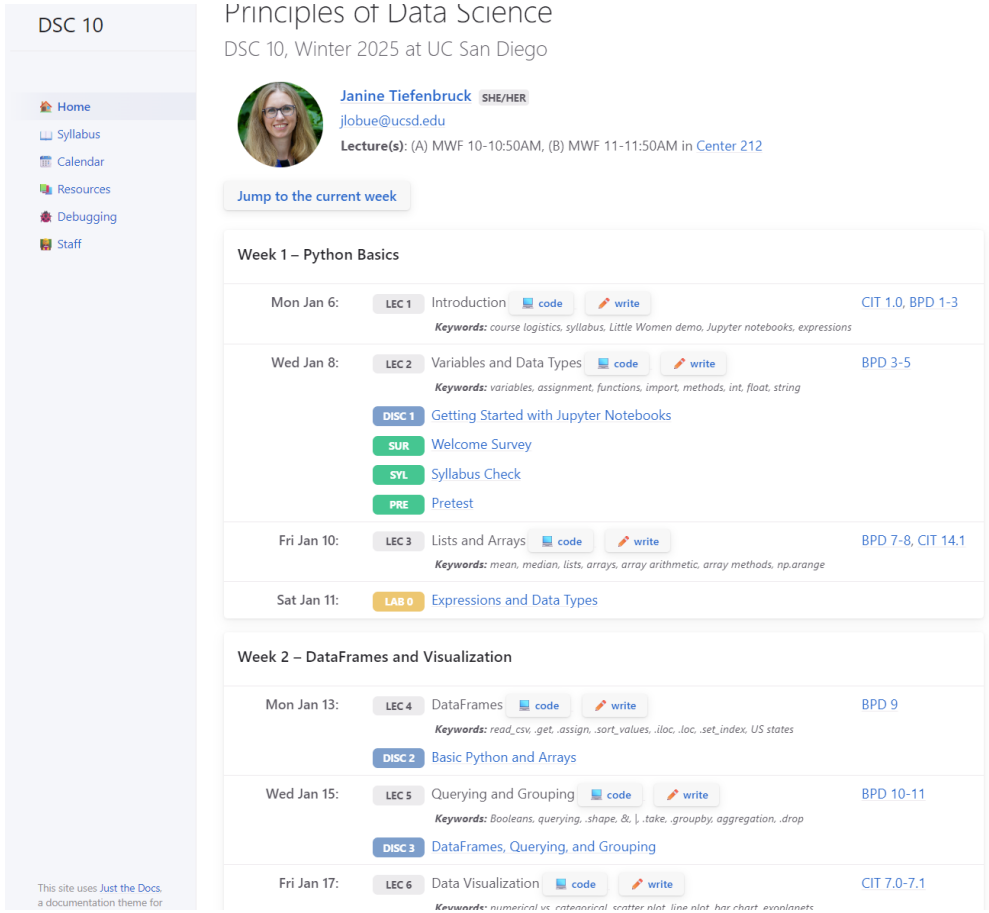
This book is an introduction to programming in C, and system programming (processes, threads, synchronization, networking and more!). We assume you've already had some programming experience, in an earlier computer science course. If you have any typos to report or content to request, feel free to file an issue at the link above. Happy Reading!

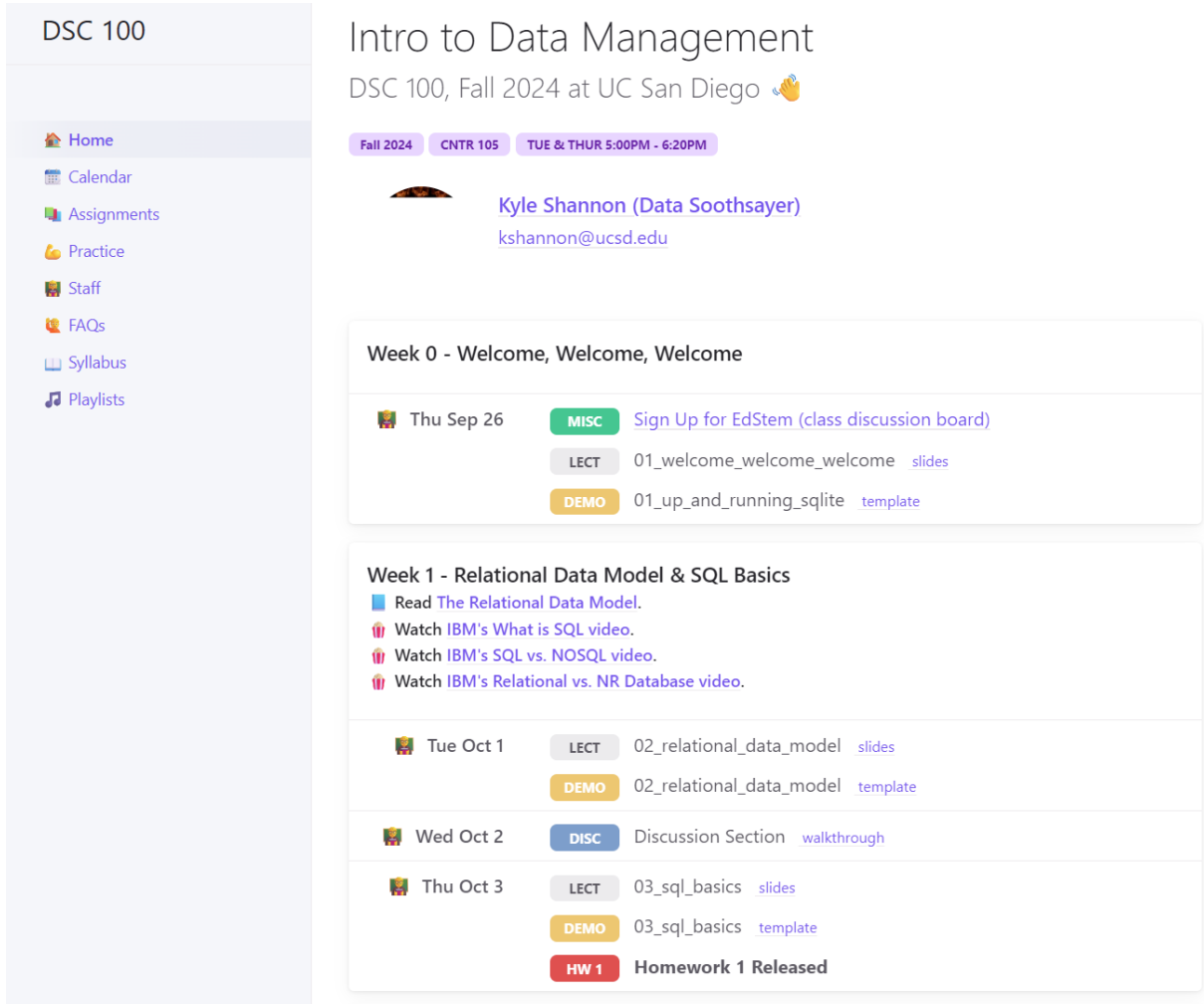
One Big PDF PDF

# MIT 18.404J: Theory of Computation

| SES # | TOPICS | KEY DATES |
|---|---|---|
| 1 | Introduction, Finite Automata, Regular Expressions | |
| 2 | Nondeterminism, Closure Properties, Regular Expressions → Finite Automata | |
| 3 | The Regular Pumping Lemma, Finite Automata → Regular Expressions, CFGs | |
| 4 | Pushdown Automata, CFG ↔ PDA | |
| 5 | The CF Pumping Lemma, Turing Machines | Homework 1 due |
| 6 | TM Variants, the Church-Turing Thesis | |
| 7 | Decision Problems for Automata and Grammars | |
| 8 | Undecidability | |
| 9 | Reducibility | Homework 2 due |
| 10 | The Computation History Method | |
| 11 | The Recursion Theorem and Logic | |
| 12 | Time Complexity | Homework 3 due |
| 13 | Midterm Exam | |
| 14 | P and NP, SAT, Poly-time Reducibility | |
| 15 | NP-Completeness | |
| 16 | Cook-Levin Theorem | Homework 4 due |
| 17 | Space Complexity, PSPACE, Savitch's Theorem | |
| 18 | PSPACE-Completeness | |
| 19 | Games, Generalized Geography | |
| 20 | L and NL, NL = coNL | Homework 5 due |
| 21 | Hierarchy Theorems | |
| 22 | Provably Intractable Problems, Oracles | |
| 23 | Probabilistic Computation, BPP | |
| 24 | Probabilistic Computation (cont.) | |
| 25 | Interactive Proof Systems, IP | Homework 6 due |
| 26 | coNP ⊆ IP | |
| 27 | Final Exam | |

# UCSD DSC10 + DSC100 : Data Science + Databases



# Alberta: Software Design and Architecture

## Syllabus

### Course 1 - Object-Oriented Design

This course takes Java beginners to the next level by covering object-oriented analysis and design. You will discover how to create modular, flexible, and reusable software, by applying object-oriented design principles and guidelines. And, you will be able to communicate these designs in a visual notation known as Unified Modelling Language (UML).

You will be challenged in the Capstone Project to apply your knowledge of object-oriented design by evolving and documenting the Java codebase for an Android application with corresponding UML documentation.

### Course 2 - Design Patterns

This course takes Java beginners to the next level by covering object-oriented analysis and design. You will discover how to create modular, flexible, and reusable software, by applying object-oriented design principles and guidelines. And, you will be able to communicate these designs in a visual notation known as Unified Modelling Language (UML).

You will be challenged in the Capstone Project to redesign an existing Java-based Android application to implement a combination of design patterns. You will also critique a given Java codebase for code smells.

### Course 3 - Software Architecture

The way that software components — subroutines, classes, functions, etc. — are arranged, and the interactions between them, is called architecture. In this course you will study the ways these architectures are represented, both in UML and other visual tools. We will introduce the most common architectures, their qualities, and tradeoffs. We will talk about how architectures are evaluated, what makes a good architecture, and an architecture can be improved. We'll also talk about how the architecture touches on the process of software development.

In the Capstone Project you will document a Java-based Android application with UML diagrams and analyze evaluate the application's architecture using the Architecture Tradeoff Analysis Method (ATAM).

### Course 4 - Service-Oriented Architecture

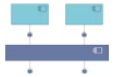
Based on an understanding of architectural styles, you will review architectures for web applications, then explore the basics of Service-Oriented Architecture (SOA) in two approaches: Web Services (WS*) and Representational State Transfer (REST) architecture.

In the Capstone Project you will connect a Java-based Android application with Elasticsearch, a web service with a REST application programmer interface (API).

**Object-Oriented Design**
Course 1 • 17 hours

**Design Patterns**
Course 2 • 15 hours

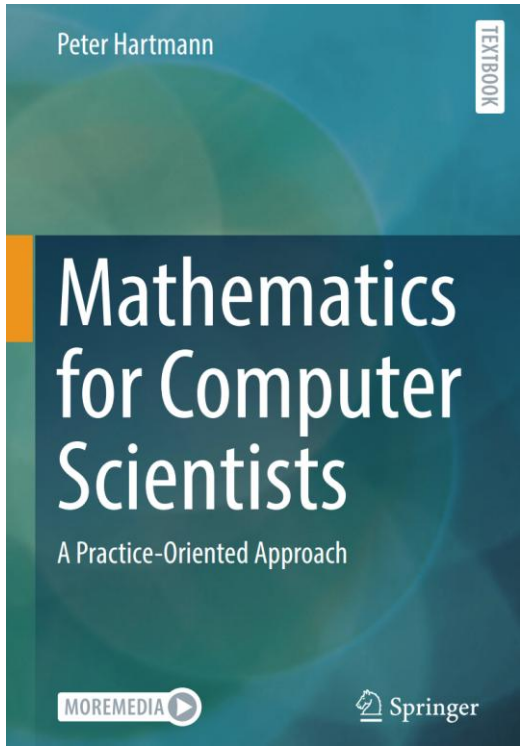
**Software Architecture**
Course 3 • 9 hours

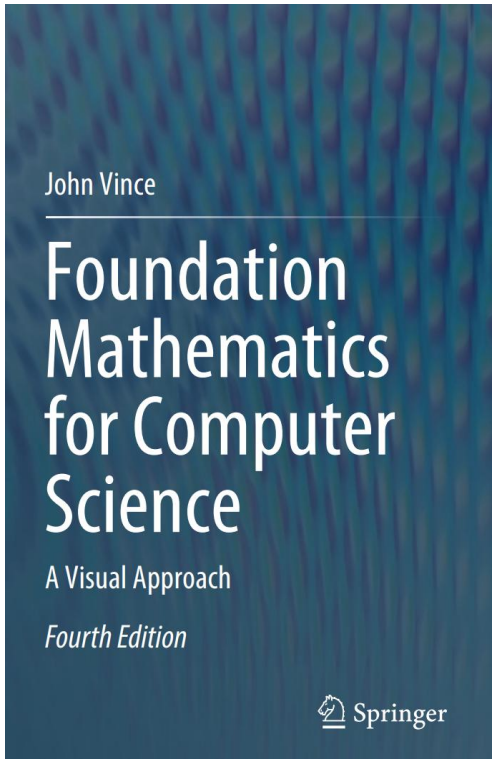


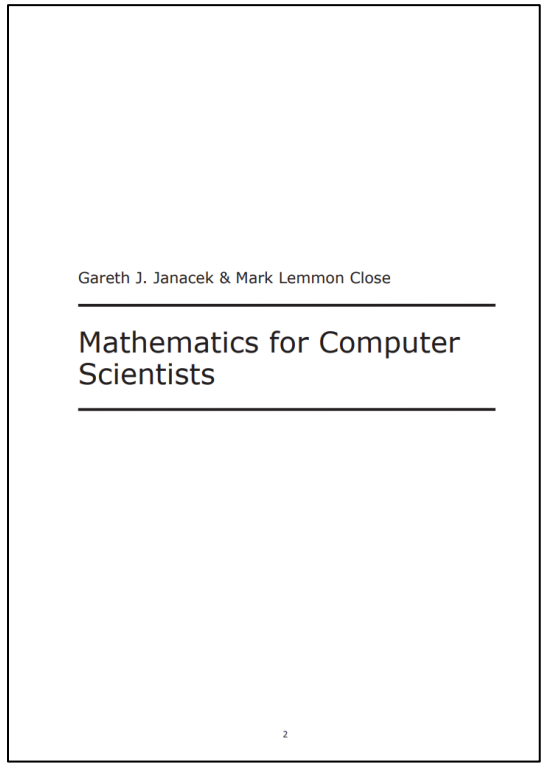
**Service-Oriented Architecture**
Course 4 • 9 hours

Alexander(Sasha) Apartsion. 28/04/25

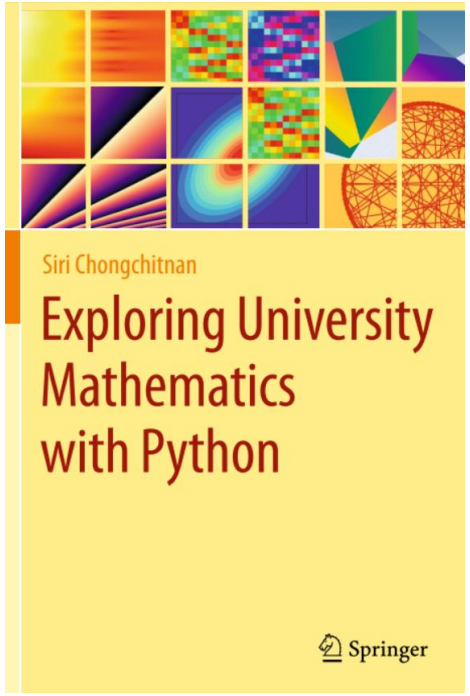
# 8.1 CS-Oriented Mathematics

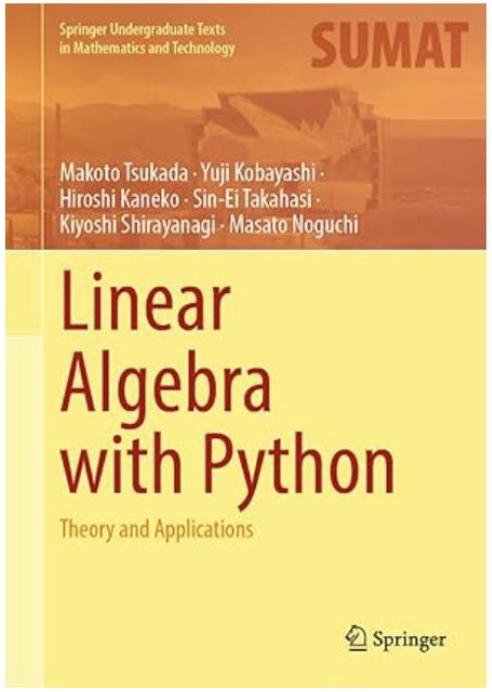


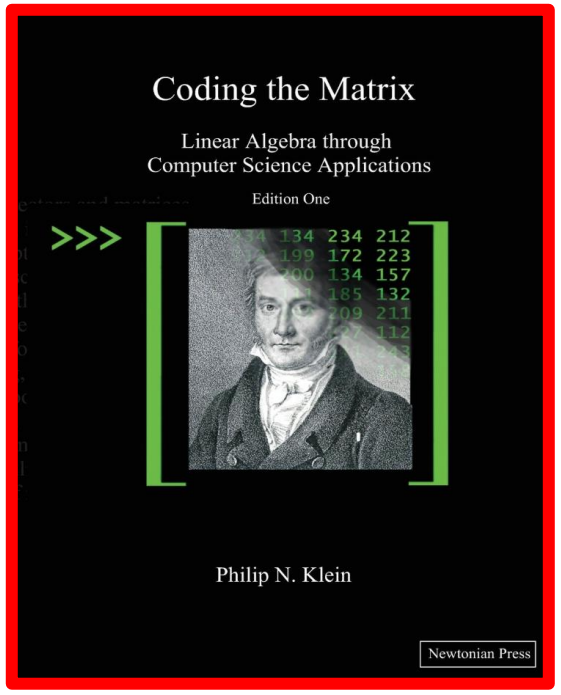



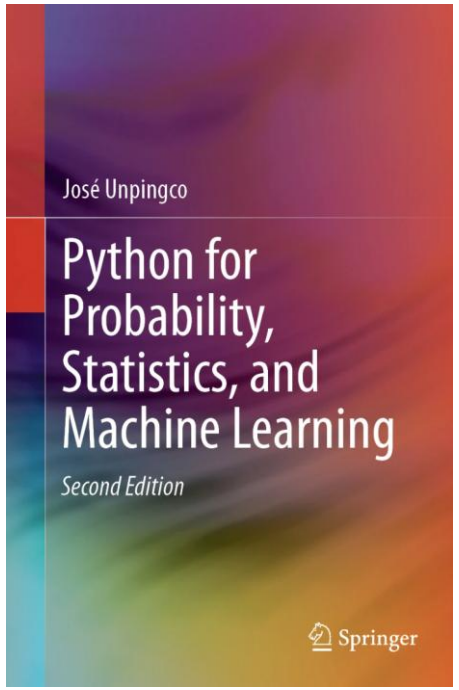
# 8.2 Code-Oriented Math Textbooks



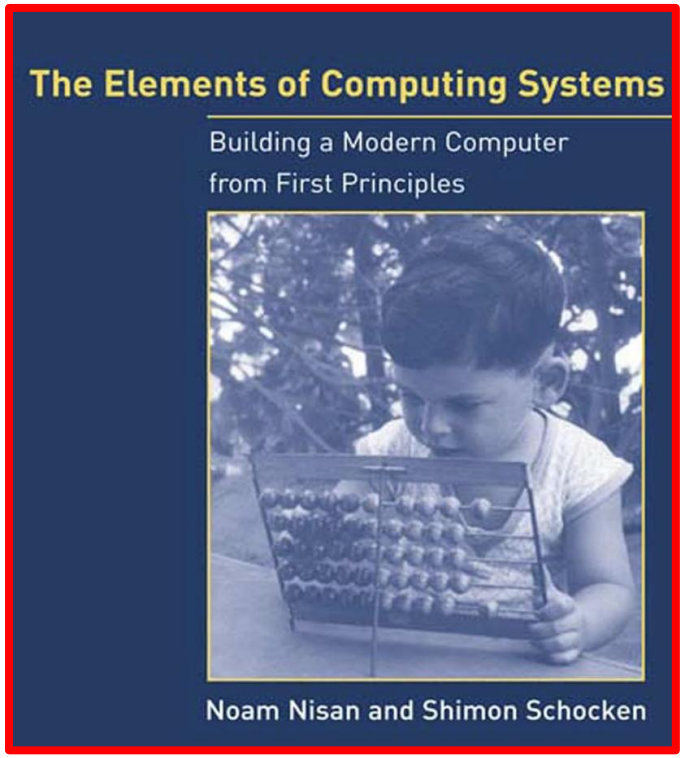



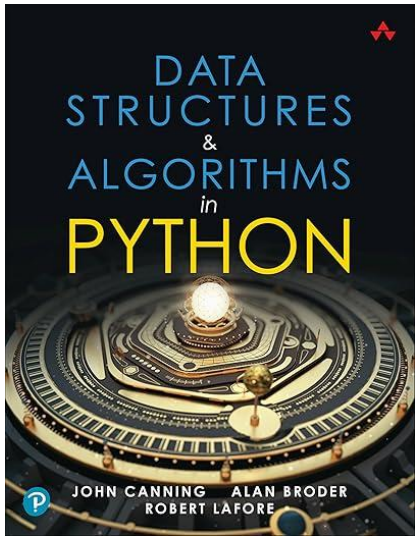



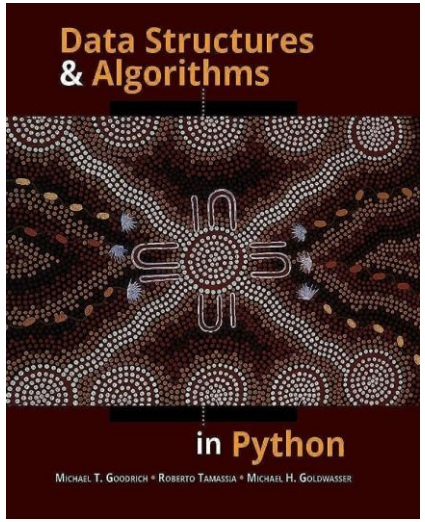

https://codingthematrix.com/

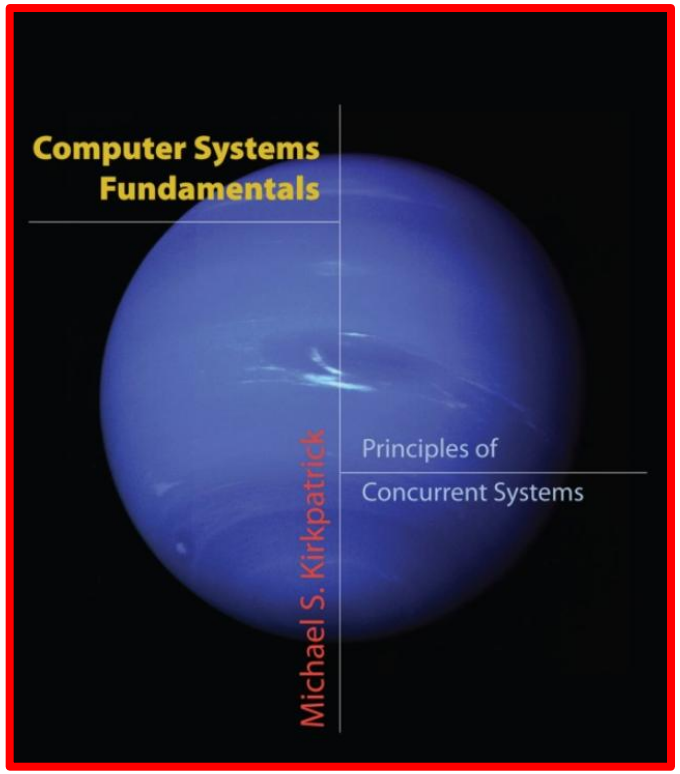


# 8.3 Hands-on core CS



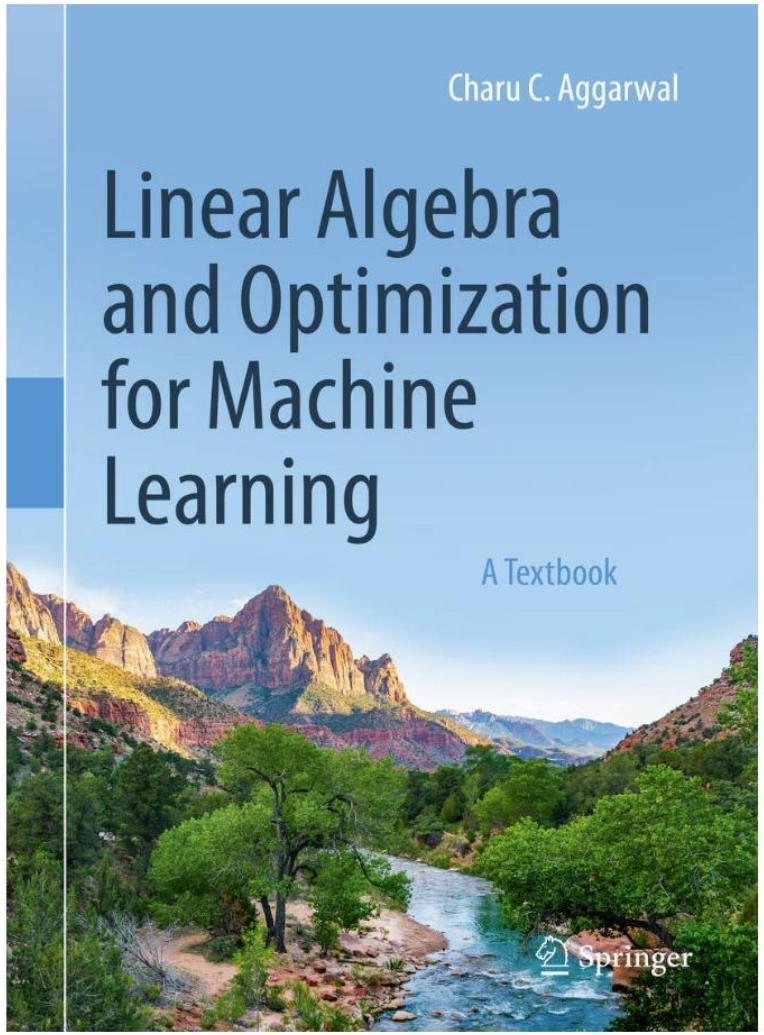

https://www.nand2tetris.org/

HW+SW(Basics of OS/Compiler)

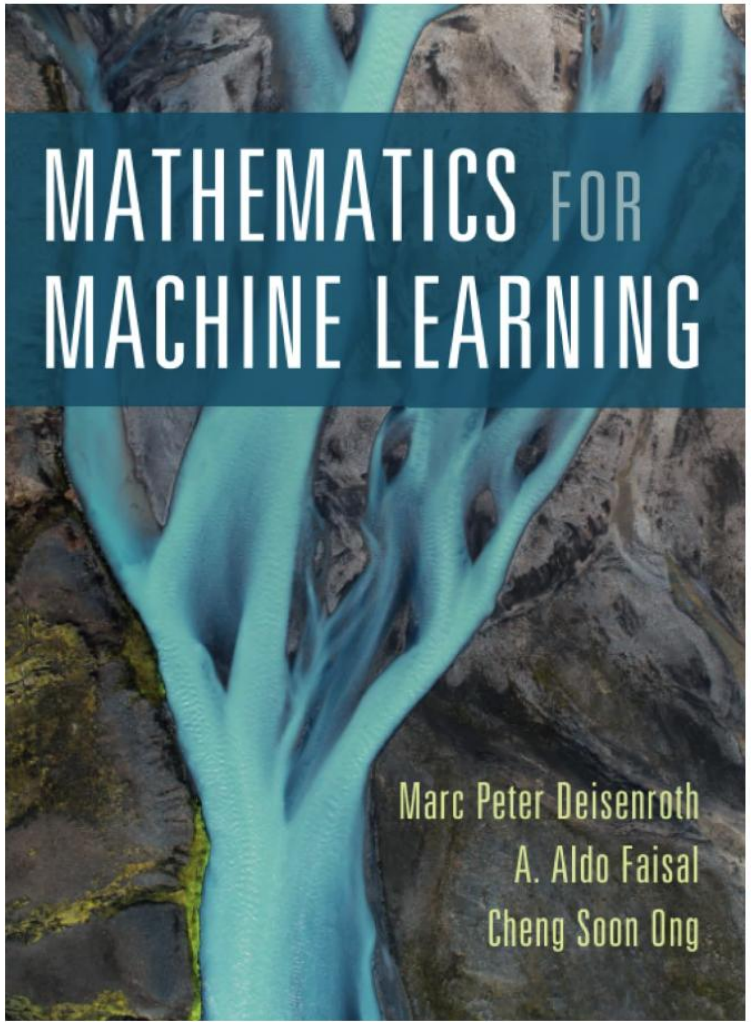
https://opencsf.org/Books/csf/html/index.html

OS + Networking

# 8.4 Mathematics with use cases for ML and AI

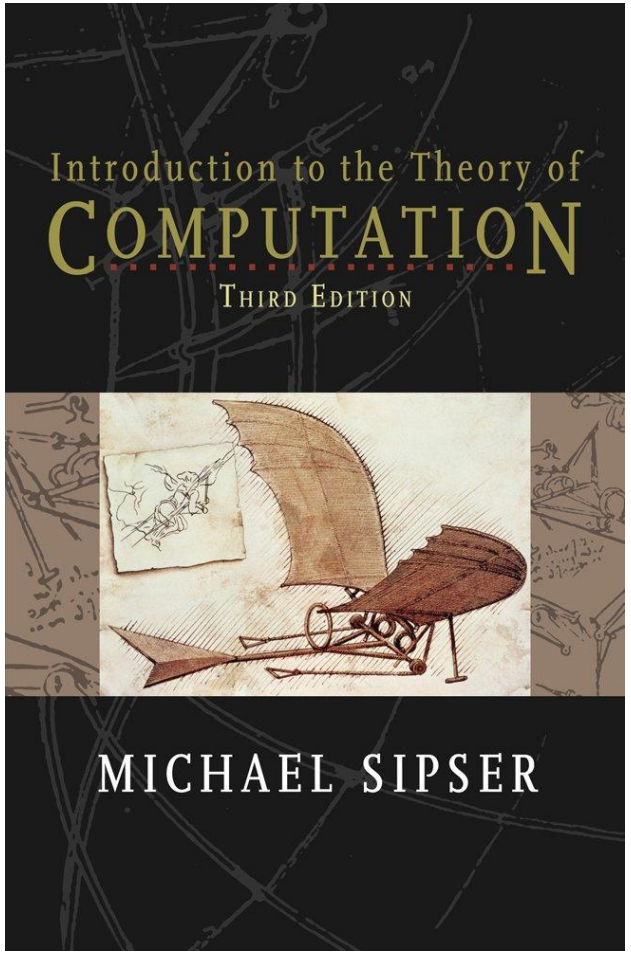



Book

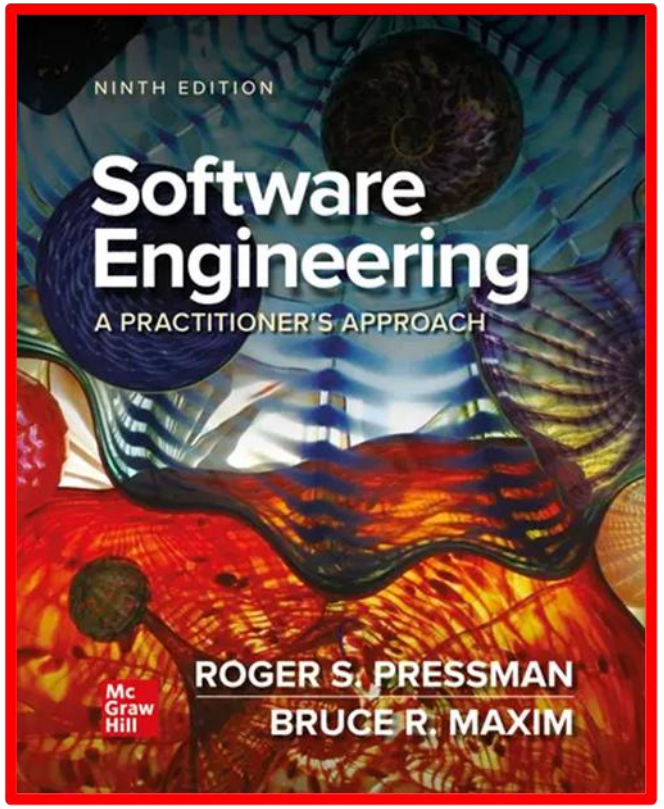


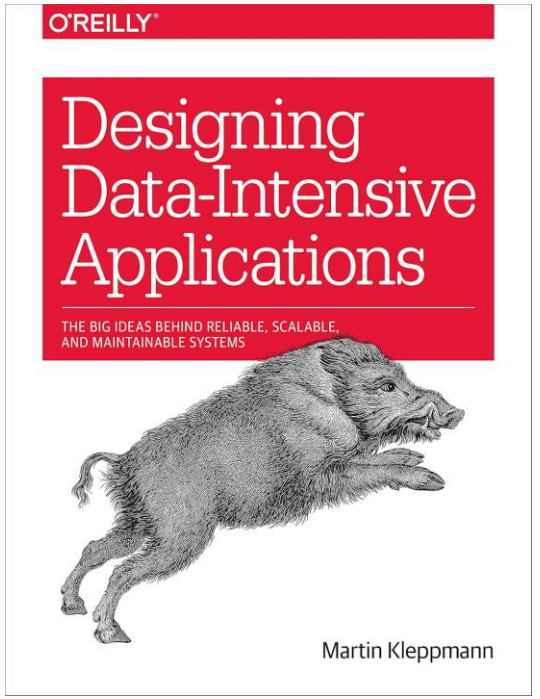

Book and material

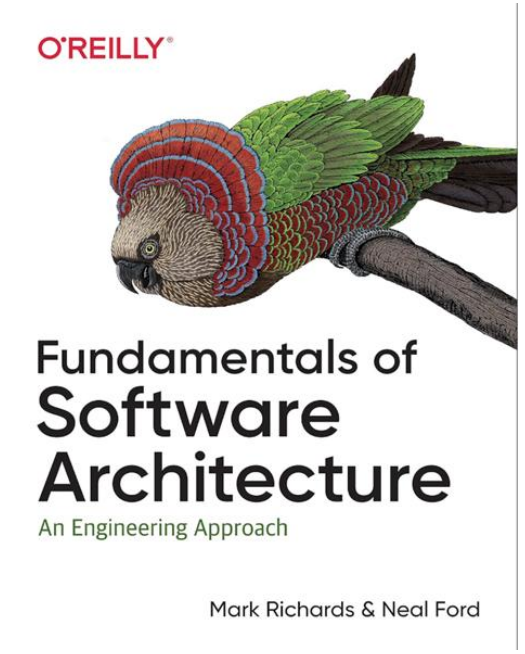
# 8.5 Advanced CS Courses




[MIT]



# No Base courses yet: Requirements

- Advanced Algorithms
  - Number theory algo (prep to Crypto/Cyber specialization)
  - Randomized algorithms (prep to Crypto/Cyber specialization)
  - Combinatorial Optimization/Computation Geometry/A* Search (prep to Robotics/AI)
  - https://web.stanford.edu/~ashishg/cs261/
  - https://ocw.mit.edu/courses/6-046j-design-and-analysis-of-algorithms-spring-2015/pages/calendar/
- Software Engineering
  - Include all phases of SDLC: Requirements/Design/Deployment
  - Techniques: waterfall/agile
  - Tools usage: DevOps, CI/CD, automating testing
  - Developing secure Software
  - Include AI-driven SE
  - *Practical Project*
  - Candidate: https://www.coursera.org/specializations/software-development-lifecycle
- Machine Learning
  - Good Classical coverage
  - Some Deep introduction
  - Highlights of Generative AI