\WOOLF/

BSc in CS Program Handbook

Date of Publication: August 2023

Foreword

This manual has been designed to familiarize you with the policies and procedures that shape the Woolf BSc in CS Program. This manual should not be viewed as complete and is not designed to replace the Woolf Academic Handbook. It is intended to provide information you will need in order to make decisions about your graduate studies and to acquaint you with the administrative requirements, policies and procedures you will be expected to meet that are outside the scope of the Woolf Academic Handbook. This document should thus be used in tandem with the Woolf Academic handbook. Where either manual seems incomplete, you are encouraged to inquire with your Faculty Advisor. For questions beyond the domain of your Advisor you are encouraged to reach out to help@woolf.university.

We hope that your experience within Woolf's program will be fulfilling, stimulating, and engaging. We are excited to welcome you to the Woolf community.

The Mission of Woolf

Woolf exists to promote academic excellence, broaden access to higher education, and guard values that are humane, democratic, and international. Above all, Woolf values freedom of thought, freedom of inquiry, and freedom of expression.

We do this through our commitment to high-quality education and through widening the horizon of opportunity by connecting students with quality academics across the world.

Through Woolf's world-class platform and programs students gain exposure to new ideas, new ways of understanding, and new ways of learning. By uniting exceptional faculty with motivated students, Woolf is able to build an outstanding academic community leading to journeys of intellectual transformation. From this we hope that students will share their academic, intellectual and other talents in serving their communities across the world.

Woolf is guided by the following tenets:

Background. Talent may be evenly distributed but opportunity is not – we are working to widen the horizon of opportunity by connecting students and teachers across the world.

Education. Woolf prioritizes an education that will serve its students both in the near-term and in the long-term. Woolf seeks to provide a personalized, bespoke education. In all fields, Woolf seeks to instill values of curiosity, intellectual discipline, and clarity of expression.

Research. Woolf prioritizes research-driven teaching that uses the latest academic scholarship, and Woolf encourages its students to engage in groundbreaking research.

Society. Woolf encourages partnerships with governments, educational institutions, research centers, schools, and businesses of all kinds – provided these partnerships do not infringe on the values of Woolf.

Technology. To the extent that existing or new technologies can improve the educational outcomes for students, widen access to the Woolf global network, improve the career experience of academics, better secure credible governance, lower the costs of institutional management, and generally support the mission of Woolf – these are embraced.

In all things, Woolf values excellence and measures itself against the highest international standards. Woolf seeks to raise those standards further.

Academic Program Information

Program Description

The Bachelor of Science in Computer Science (BS in CS) is a 120-credit program that includes a selection of online courses. The program is an integrated, sequential course of study in which students obtain and demonstrate the knowledge and skills required of the computer science industry.

The BSc in CS teaches students comprehensive and specialized subjects in computer science; it develops skills in critical thinking and strategic planning for changing and fast-paced environments, including technological and operational analysis; and it develops competences in leadership, including autonomous decision-making, and communication with team members, stakeholders, and other members of a business.

Courses include instruction targeting the areas above as well as and other relevant learning goals, opportunities for repeated practice, and methods for students to demonstrate their accomplishment of the outcomes.

The BSc in CS is delivered completely digitally by combining asynchronous components (lecture videos, readings, and written assignments) and synchronous cohort meetings attended by students and an instructor or faculty member during a video call.

The asynchronous components support the schedules of students from diverse work-life situations, and synchronous meetings provide accountability, motivation, and a sense of community presence for students. The synchronous sessions allow unparalleled access to high quality instruction and enhanced collaboration among students through using face-to-face online interaction.

Faculty conduct live office hours with students and interaction between faculty members and students, both individually and as a group, is enhanced in the online environment by blending asynchronous content with real-time student responses. Faculty and enrolled students have 24/7/365 access to technical support through Woolf's support system.

Woolf's digital campus allows students to complete the program in as little as 3 years of continuous study or within 6 years as part of a part-time course of study. The degree on

the students' transcripts is a Bachelor of Science in Computer Science, which attests to their completion of the requirements.

Admission Requirements

The Woolf BSc in CS is a fast-paced, rigorous degree focused on teaching comprehensive and specialized subjects in computer science. Candidates should have a high school diploma (or equivalent).

English language competency at an IELTS 6 or equivalent is required of all applicants.

The program is suited for undergraduate students considering a career in technology or the innovation (start-up) economy. The overall programme is designed for those with little or no background in computer programming and only basic mathematical knowledge is required.

In all cases, the target group should be prepared to pursue substantial academic studies.

Curriculum Areas

The program is organized into a course structure of three tiered areas. Each tiered area sequentially builds off of the previous, so students must complete each tier before advancing to the next.

Each course consists of regular lessons and cumulative lessons devoted to cumulative examination. Each course requires about 150 hours to complete (see individual courses for details). A full-time student completes two lessons per week with an assignment submitted for each lesson; this pattern continues for each regular lesson in the class.

Summative examination lessons allow an appropriate amount of time for students to review and revise their prior work and deepen their synthetic grasp of the materials in preparation for cumulative examination or project.

Learning Outcomes

The program teaches students comprehensive and specialized subjects in computer science; it develops skills in critical thinking and strategic planning for changing and fast-paced environments; and it develops competences in leadership, including autonomous decision-making, and communication with employers, stakeholders, and other members of a team.

Knowledge

- Students will grasp major concepts of computer science and web engineering, and be able to classify specific computer science issues and engineering tasks as instances of broader principles and generalizations.
- When completing assignments, students will demonstrate an understanding of advanced general computer science concepts and will be able to use terminology from the domain correctly, and they will rely on specific facts, including those at the forefront of their field of study.
- Students will be able to contextualize factual knowledge of computer science issues in view of relevant social and ethical issues.

- Students will display creative thinking on the basis of the knowledge they gain in the course in response to concrete and abstract problems.

Skills

- Students demonstrate some application of theoretical and practical knowledge in responding to problems.
- Students formulate their ideas in clearly structured conventional formats and use appropriate evidence to support their claims.
- Students will monitor, evaluate, and adjust their own learning needs in order to succeed as independent learners.
- Students will also collect and analyze data to respond to both well-defined practical problems and well-specified abstract problems.

Competences

- Students will manage well-defined IT projects with a range of responsibilities that require independent decision-making and handling of unpredictable situations.
- Students will gain professionalism, discipline, and creativity through managing projects and collaborating with others.
- Students will develop the learning skills needed to continue to undertake further, self-directed studies in computer science and programming with a high degree of autonomy.

Faculty

All instruction is provided by competent academics with qualifications commensurate to their role. All teachers are also expected to have relevant teaching experience in the domain of their expertise. All faculty members at Woolf are expected to be in possession of a research doctorate in the domain of their teaching or supervision; moreover, they are expected to have a record of research or a research agenda reflecting the capacity for research.

Woolf uses clear, fair, and transparent processes for teaching recruitment, conditions of employment, and professional advancement. Notices of availability are publicly listed on the Woolf platform and, when available, other sites visited by academics. Criteria for teaching positions, including any associated conditions of ongoing employment, are clearly stated. Applications for teaching are reviewed by the Administrative Board, or a committee of the Board, until a position(s) is filled. Notices state the supporting documentation required as evidence for the review of an applicant. All applicants are required to demonstrate their competence for the teaching position by providing a copy of their credentials to be verified before the position is filled. This policy applies to all teaching roles of Woolf Education Ltd, including any teaching services provided by third party vendors, which are subject to the same process of review. In all cases, the final decision for filling a role in accordance with the criteria stated on the public notice is made by the Administrative Board.

Woolf's policies and procedures apply consistently to full-time, part-time, *ad hoc*, and third-party teaching activities. All teaching activities fall within the scope of Woolf policy. Teaching staff, including part-time or *ad hoc* teaching staff are directed towards updates and developments in their field as well as the methodological requirements for their programs.

All Faculty Members are encouraged to discuss innovative forms of teaching, formulate how these may be implemented, and propose those implementations in the Faculty Council. At the end of all courses, students provide feedback on their learning experience, and twice per year faculty provide feedback by survey. All teachers are expected to maintain a record of student outcomes, and teaching activities are periodically reviewed or observed. In cases of disagreement, or suspected misconduct, fraud, or prejudice, a Red Flag should be submitted under the Red Flag Procedure.

All courses and programs are subject to processes of quality enhancement to improve student outcomes, including the course's continued review to assess its scholastic rigor and value.

Faculty Requirements

Academic Staff are called Faculty members at Woolf. Faculty members at Woolf must possess a research doctorate and are expected to have a record of peer-reviewed research. All teaching is under the authority and oversight of a faculty member – including instructional design, synchronous meetings, and lectures. In cases where pre-recorded lectures or podcasts are provided that contain content from outside of Woolf, any such content is to be produced by lecturers who are experts with a research doctorate in the relevant domain, or where relevant, by those with at least 7 years of industry-specific experience.

Teaching Staff are called Expert Instructors, Domain Experts, or simply Instructors. Expert instructors are used in courses to provide domain-specific industry insights, including insights and feedback on student work during synchronous meeting sessions. Expert instructors must have management experience and be in possession of at least a master's level qualification. Expert instructors are under the direct authority of the Faculty Members and must be trained in Woolf's pedagogical methods.

Faculty Advisors

Colleges at Woolf exist to support their members and provide helpful resources to students.

In the tradition of Harvard's "Houses" and Oxford's "Colleges", Woolf provides every student with membership in a Woolf college for support during their academic journey. Every student should be assigned a Faculty Advisor, who is a faculty member from within the student's own college, and who acts as the first point of contact for non-technical academic issues related to the student's progress, particularly where these may benefit from an independent point of view. Students are strongly encouraged to meet with their advisors at least once each semester.

Thus every faculty member oversees their own registered students through the normal synchronous teaching sessions, and office hours, and additionally provides availability that can be booked for advisees, should the need arise.

Program Outline

Outline of Program		
Course Title	**Credits**	**Course Description**
1. Introduction to Programming in Python	1.5	This course is intended for students with little or no programming experience. It aims to help students develop an appreciation for programming as a problem solving tool and to provide a foundation in Python programming. Students will learn how to think algorithmically, solve problems efficiently, and prepare for further computer science studies.

2. Optimize Your Learning	1.5	During the course, students will develop competence in skills that are most critical for effective self-directed and self-regulated learning (i.e. self-management, self-monitoring, and self-modification), while also learning how to use learning strategies to maximize their overall learning efficiency and efficacy. They will also utilize the Emotional Intelligence framework to explore their identity, self-image, motivation, and self-regulation skills, to support their development as self-directed learners.
3. Web Foundations	1.5	This course provides a foundation in building for the web. It helps students understand how the internet works, examines the role of the internet in their lives, and teaches them the basics of web development. The course prepares students for the advanced course in Web Application Development.
4. Programming 1	3	The course helps students develop an appreciation for programming as a problem-solving tool. It teaches students how to think algorithmically and solve problems efficiently, and serves as the foundation for further computer science studies.

5. Mathematical Thinking	3	This course helps students develop the ability to think logically and mathematically. It prepares students for more advanced courses in algorithms and discrete mathematics. An emphasis is placed on the ability to reason logically, and effectively communicate mathematical arguments.
6. Collaborating for Impact	3	The course will start by focusing on the social awareness and relationship management components of Emotional intelligence, which were briefly introduced in the pre-requisite course, Optimizing Your Learning. Students will then be introduced to a variety of collaboration and leadership theories and frameworks, and they will examine theories of team dynamics and dysfunction, and reflect on how these relate to their past experiences of collaboration in academic and personal settings.
7. Communicating for Success	3	Communicating for Success supports students in developing communication skills that are essential for success in their personal and professional lives. The course will focus on close reading, written communication, verbal communication, and non-verbal communication skills. An emphasis will be placed on weekly submissions, and peer and instructor feedback, to allow students to practice and improve their skills.

8. Programming 2	3	This course expands on Programming 1, and deepens students' knowledge of Python with a focus on data access and management.
9. Web Application Development	3	This course builds on Web Foundations, and provides a comprehensive introduction to client and server-side development for the web.
10. Front End Web Development	3	Front End Web Development builds on previous knowledge of web development, and extends students' familiarity with modern HTML, CSS, JavaScript, and Web APIs. Students learn to develop and deploy client-side web applications with greater scope and complexity. Complex frontend features require using HTML, CSS, and JavaScript together. Students will usually have taken Web Application Development (or similar course under advisement from their faculty) as a prerequisite for this course.

11. Data Structures and Algorithms 1	3	This course teaches the fundamentals of data structures and introduces students to the implementation and analysis of algorithms, a critical and highly valued skill for professionals.
12. Product Management and Design	3	This course teaches students to build products users want and love. It gives students a foundation in the tools and practices of modern product management and interaction design. Students will work in pairs to apply product development skills to real user challenges.
13. Team Software Project	3	In this course, students practice the skills necessary to work effectively on a professional software product team. By working in small teams to build a web application, they integrate the technical, communication, and collaboration skills built in previous courses.

14. Industry Experience 1	6	Industry Experience is a form of experiential learning that enables students to apply their academic knowledge in a professional context. Students work to build software that meets the needs of a professional organization by completing either (1) an approved internship, or (2) a product studio.
15. Programming in Python	3	This course is intended for students who have completed Programming 1. It aims to help students deepen their programming skills, concentrating on Python. Students will learn how to think algorithmically and solve problems efficiently.
16. Engineering for Development	3	Engineering for Development, Challenge Studio 1, and Challenge Studio 2 are courses that help students investigate the role that technology can play in solving some of the world's most intractable social and economic development challenges.

17. Discrete Math	3	This course builds on Mathematical Thinking and provides the mathematical foundation needed for many fields of computer science, including data science, machine learning, and software engineering.
18. Computer Systems	3	This course explores computing beyond software. Students will go a level deeper to better understand the hardware and see how computers are built and programmed. It is modeled on the popular, project based “Nand to Tetris” textbook, which walks learners through building a computer from scratch. It aims to help students become better programmers by teaching the concepts underlying all computer systems. The course integrates many of the topics covered in other computer science courses, including algorithms, computer architecture, operating systems, and software engineering.
19. Challenge Studio 1	3	Engineering for Development, Challenge Studio 1, and Challenge Studio 2 are 3 courses that help students investigate the role that technology can play in solving some of the world’s most intractable social and economic development challenges.

20. Data Structures and Algorithms 2	3	This course builds on Data Structures & Algorithms 1. Students will explore non-linear data structures, and implement and analyze advanced algorithms.
21. Introduction to Data Science	3	Data science is applicable to a myriad of professions, and analyzing large amounts of data is a common application of computer science. This course empowers students to analyze data, and produce data-driven insights. It covers all areas needed to solve problems involving data, including preparation (collection and integration), presentation (information visualization), analysis (machine learning), and products (applications).
22. Challenge Studio 2	3	Engineering for Development, Challenge Studio 1, and Challenge Studio 2 are 3 courses that help students investigate the role that technology can play in solving some of the world's most intractable social and economic development challenges.

23. Network and Computer Security	3	Network and Computer Security teaches students the principles and practices of security for software, systems, and networks. It aims to make students critical examiners and designers of secure systems. Students will learn the mathematical and theoretical underpinning of security systems, as well as practical skills to help them build, use, and manage secure systems.
24. Industry Experience 2	6	Industry Experience 2 provides a form of experiential learning that enables students to apply their academic knowledge in a professional context. Students work to build software that meets the needs of a professional organization by completing either (1) an approved internship, or (2) a product studio.
25. Backend Development	3	Back End Development builds on previous knowledge of web development and security, and equips students with knowledge of server-side development so that they can become professional back-end developers and build enterprise-scale applications. Students learn to develop and deploy server-side applications with greater scope and complexity.

26. iOS App Development	3	iOS App Development teaches students to build modern mobile applications using Apple's iOS development platform and tool chain. It prepares students to become professional iOS developers. Students learn the core principles of the Swift programming language and Apple's front-end frameworks for creating single and multi-page applications.
27. Ethics for Tech	3	This course examines the ethical questions that are emerging as a result of rapid technological change. It prepares students to become responsible technologists, and provides a basis for ethical decision-making in their professional work.
28. Android App Development	3	Android App Development teaches students to develop applications for Android, the most-used mobile platform in the world. Students learn to build with Android Studio and Kotlin, the modern toolkit used for professional Android development. The course prepares students to become professional Android developers.

29. Artificial Intelligence	3	Artificial Intelligence (AI) aims to teach students the techniques for building computer systems that exhibit intelligent behavior. AI is one of the most consequential applications of computer science, and is helping to solve complex real-world problems, from self-driving cars to facial recognition. This course will teach students the theory and techniques of AI, so that they understand how AI methods work under a variety of conditions.
30. Machine Learning	3	This course aims to teach students the theoretical and practical methods for solving problems using machine learning. Machine learning is one of the fastest-growing areas of computer science. Its applications are reshaping society, from consumer products (e.g., voice assistants and recommendations) to life-sciences (e.g., personalized medicine and tumor detection). This course will build on the Data Science introductory course, and help students understand how, why and when machine learning methods work.
31. Interaction Design	3	This course introduces students to the principles of human-computer interaction (HCI). Students explore how humans process information (perception, memory, attention) in the context of designing and evaluating interfaces. This course complements programming coursework by helping students understand how to design more usable systems.

32. Designing Your Future	1.5	Designing Your Future is inspired by the Stanford University course Designing Your Life. In this course, final year students will use design thinking to reflect on their undergraduate studies, and to plan and prepare for their transition into full-time employment
33. Capstone	3	The Capstone Research Methods course supports students in developing critical research skills that are needed for the successful completion of their capstone project (Applied Computer Science).
34. Applied Computer Science	7.5	This capstone course enables students to demonstrate their proficiency in the technical and human skills that they have acquired throughout their undergraduate studies. The capstone requires students to conceptualize, plan, and implement a software project to completion, and evaluate their project's processes and outcomes.

Assessment and Grading

General Procedures

Academic assessment at Woolf is of two kinds: regular and cumulative. Regular assessment applies to the continuous evaluation of student progress, concentrating on the

proficiency of submitted assignments, and the ability of the student to respond to issues raised by the instructor during an instructional session. Cumulative assessment applies to the final project assignment. This requires the students to deepen and extend the scholarly engagements initiated in their prior work.

Students who fail any one course of the degree, cannot progress to complete the degree, except by approval of the College Dean or College Academic Committee. Failed courses may be retaken at the approval of the College Dean or College Academic Committee, or by appeal, at the discretion of the Quality Assurance, Enhancement, and Technology Alignment Committee (QAETAC). For more information about QAETAC please see the Woolf Academic Handbook.

Cumulative Examination of Courses

Traditionally, cumulative examination of courses is by a submitted final project in the form of a long assignment.

The long assignment is meant to synthesize, deepen, and extend the learning outcomes of the regular lessons while introducing new material and insights. Not more than 50% of the long assignment may be material taken from other assignments. The topic of the assignment must be agreed in advance with the instructor. Examination assignments are expected to be completed at a high research standard, and must be well-structured, well-crafted, and contain appropriate citations to the primary and secondary literature of the course.

Mode of Teaching and Assessment

The Online Campus

Woolf University's courses are offered almost exclusively through its proprietary learning platform. The platform supports both asynchronous and synchronous modes of learning. The asynchronous portion of programs includes structured course materials that the course lead and course instructors prepare ahead of time. These courses are done independently of students' classmates and according to the student's own schedule, but prior to the synchronous sessions. Synchronous sessions are held in the "virtual classroom," where students and faculty use internet technology such as video conferencing and web cameras to ensure they are actively engaged in the learning process. It is essential for students to connect with each other, share information, and create professional networks and relationships as they would in a traditional program or within a professional setting. For information about the technical specific requirements please see the Technical Requirements section within the Woolf Academic Handbook.

Structure of the Courses

The bSc in CS combines asynchronous components (lecture videos, readings, and written assignments) and synchronous meetings attended by students and an instructor or faculty member during a video call.

Asynchronous components support the schedules of students from diverse work-life situations, and synchronous meetings provide accountability and motivation for students.

Each short foundation course, and the extended specialist courses, consists of regular lessons and cumulative lessons devoted to summative examination.

As is typical for synchronous teaching, the student will compose one assignment (such as a report, 1,000 word essay, financial model, or presentation) per lesson, which is the topic of meeting discussion. A full-time student completes two lessons per week with an assignment submitted for each lesson; this pattern continues for each regular lesson in the course.

Summative examination lessons allow an appropriate amount of time for the student to review and revise his or her prior work and deepen their synthetic grasp of the materials in preparation for cumulative examination or project.

Contact Hours

For the breakdown of hours, consult the Pedagogical Procedures and Assessment sections.

Students engage in 8 synchronous meeting sessions in which questions and answers about asynchronous study materials are addressed.

Synchronous meeting engagements include not only 60-75 minutes of intensive contact time between the students and faculty member in every lesson (up to twice per week), but also extends beyond the synchronous meeting session itself to include ongoing supervisory support on an ad hoc basis (typically by email or brief virtual meeting). Students are closely supervised by asynchronous direction, oversight, feedback, and guidance; and occasionally new handouts or other scholarly materials are provided to follow up on issues raised in a synchronous meeting discussion. Thus, on our calculation, we allocate four further hours of contact time beyond the synchronous session to capture the individual, personalized, intensive, bespoke form of guidance that a student receives. We calculate all other forms of contact time at a 1:1 ratio, including lectures, whether delivered synchronously or as pre-recorded videos or podcasts.

Example regular lesson Breakdown

* = contact hours = 66

† = assessment hours = 1.25

Hours	**Activity**
1.25*†	synchronous session
4*	Time under the direction and control of a tutor
1.5*	Lecture videos or podcasts
8.25	Independent reading & note-taking
7.5	Assignment composition
22.5	**Total**

Pedagogical Procedures

The MSc in CS will be delivered using online and blended learning techniques, which support a variety of teaching and learning methods, including the following:

- synchronous meetings;
- assigned lectures by video or podcast;
- assigned readings;
- handouts delivered electronically;
- digital material, including slideshow presentations and other assigned media provided in course packets and by weblink.

The core pedagogical method used in this course will include synchronous meetings between a faculty member, or a subject expert instructor under the oversight of a faculty member, and a small group of students. Student interaction plays a key role in the organization of each synchronous meeting, which focuses on a discussion of a student's pre-submitted assignment.

Online delivery, although identical in content for an in-person session, provides significant advantages to students in terms of accessibility – students can more easily reach academic experts across borders and more easily integrate study within the pattern of their own life or career. Further advantages of the online delivery include the digital quality assurance techniques outlined within the Academic Handbook.

Preparing for a single synchronous meeting requires about 21.25 hours. A representative workload consists of the following: students must review about 100 pages of reading material (or equivalent video content, audio content, or interactive content) and prepare a piece of written work of 2-4 pages in response to a specific set assignment question. Before the start of the synchronous session, this work is submitted to the instructor for review.

Every student must then be prepared to discuss and defend his or her written work directly with the instructor (who is an expert in the field) and the other students for up to 75 continuous minutes. Faculty members and subject experts seek to deepen the students' understanding of the material and probe aspects of the written assignment that may benefit from clarification, revision, or further exploration.

At the end of the synchronous session, the instructor provides the student with feedback on the meeting assignment, including a mark, and provides bespoke guidance for the next assignment. Students are provided with a curated reading list from the instructors, assigned pre-recorded lectures, and a research question to guide the organization of their next synchronous meeting assignment or assignment. Engaging in this activity twice per week is a full workload of about 45 hours.

The synchronous meeting system is designed to be mentally demanding and personally engaging. The pedagogical style is known for producing high-quality domain-specific learning outcomes because students must learn assigned materials and related case studies for themselves, before presenting their work to an instructor in their own words for discussion twice per week. By requiring students to describe and analyze topics in

their own words, synchronous meetings engage and extend a student's existing range of abilities.

The synchronous method is also known for producing high-quality domain-agnostic learning outcomes because students must be prepared to organize and present their perspective on an assignment twice per week, and be prepared to think analytically and creatively about what they have done. Students must learn to present their viewpoint, even while being prepared to adopt a new position in light of the evidence and under the questioning of the teacher.

Assessment

General Procedures

For the MSc in CS, assessment is of two kinds: regular assessment and summative assessment.

Regular assessment applies to the continuous evaluation of student progress, concentrating on the proficiency of submitted assignments, and the ability of the student to respond to issues raised by the instructor during a synchronous meeting session.

Cumulative assessment applies to the final project assignment. This requires the students to deepen and extend the scholarly engagements initiated in their prior work.

Students who fail any one course of the degree, cannot progress to complete the degree, and will by default fail the BSc in CS. Failed courses may be re-taken at the discretion of Woolf's Faculty Members.

Cumulative Examination of courses

For each course, a percentage of the grade derives from the average of the regular assignments, and a larger percentage of the grade derives from the cumulative examination. The cumulative examination of courses is by a submitted final project in the form of a long assignment.

The cumulative examination/long assignment is by summative assignment (3,000 word essay, or similarly-sized financial model, or presentation), which must synthesize, deepen, and extend the learning outcomes of the regular units while introducing new material and insights. Not more than 50% of the long assignment may be material taken from synchronous meeting assignments. The topic of the assignment must be agreed in advance with the faculty member and subject matter experts. Examination assignments are expected to be completed at a high research standard, and must be well-structured, well-crafted, and contain appropriate citations to the primary and secondary literature of the course.

Grading Progress

Students receive grades and feedback on each assignment throughout a course. Depending on the specific course, students may receive both a grade and a written or audio comment from an instructor. Courses display the weight assigned to each grade-bearing category (such as regular assignments, attendance, and/or final projects). Students have access to their grade book at all times and can see a record of all grades, attendance records, and assignment submissions.

The grade book displays the current running average for the grades in the course in accordance to the grade weights, inclusive of all those assignments which the student has submitted and on which the instructor has provided grades. At the end of the course, grades

are finalized and added to the student's transcript, which is accessible at all times for students enrolled in credit-bearing programs.

Grading System

The final grade for a course is determined by the weighting rules stated in the course offering. Unless otherwise stated, all courses are weighted as follows: 50% of the grade derives from the average of the instructional assignment session, and 50% of the grade derives from the cumulative examination.

The final grade on a degree is weighted in proportion to the credits of individual courses. For example, a degree composed of a 3 credit course and a 6 credit course will weigh the 6 credit courses proportionately more, according to the number of credits.

Woolf's International Grade Classification

Woolf's teachers are trained in a number of different grading scales; these scales are cross-referenced. This handbook employs the American grading system and classification, with US grades as the default.[1] US grades are the most granular and distributed with the least number of gaps, which is why we have selected them as a default marking scheme for transcripts. Woolf's international conversion scheme is as follows:

US GPA	US Grade	US Per Cent	UK Mark	UK Classification	Malta Grade	Malta Mark	Malta Classification
4	A+	97 - 100	70+	First class honors	A	80-100%	First class honors
3.8-4.0	A	94-96	67-69	Upper-second class honors	B	70-79%	Upper-second class honors
3.7	A-	90-93	65-67	Upper-second class honors			
3.3	B+	87-89	60-64	Lower-second class honors	C	55-69%	Lower-second class honors
3	B	84-86					
2.7	B-	80-83	55-59	Lower-second class honors			
2.3	C+	77-79	50-54	Third class honors	D	50-54%	Third-class honors
2	C	74–76					
1.7	C-	70–73	45-49	Third class honors			
1.3	D+	67–69	40-44	Ordinary/Unclassified			
1	D	64–66	35-39	Ordinary/Unclassified			
0.7	D-	60–63					
0	F	Below 60	Below 35		F	45-54%	

[1] Cf. the Fulbright Commission (http://www.fulbright.org.uk/going-to-the-usa/pre-departure/academics), Princeton Review (https://www.princetonreview.com/college-advice/gpa-college-admissions), European Commission (https://eacea.ec.europa.eu/national-policies/eurydice/content/second-cycle-programmes-49_en), and University of Malta (https://www.um.edu.mt/__data/assets/pdf_file/0005/47390/harmonisedregs-09.pdf).

Woolf Grading Criteria, Definition of Grades, and Classification

Grading of student work keeps in view the scale of work that the student can reasonably be expected to have undertaken in order to complete the task. The Woolf grading scheme draws heavily from the marking scheme set out by the University of Oxford (cf. History Faculty Course Handbook 2016-2018).

a. The assessment of work for the course is defined according to the following rubric of general criteria:

i. i. Engagement:

- Directness of engagement with the question or task
- Range of issues addressed or problems solved
- Depth, complexity, and sophistication of comprehension of issues and implications of the question or task
- Effective and appropriate use of imagination and intellectual curiosity

ii. Argument or solution:

- Coherence, mastery, control, and independence of work
- Conceptual and analytical precision
- Flexibility, e.g. discussion of a variety of views, ability to navigate through challenges in creative ways

iii. Evidence (as relevant):

- Depth, precision, detail, range and relevance of evidence cited
- Accuracy of facts
- Knowledge of first principles and demonstrated ability reason from them
- Understanding of theoretical principles and/or historical debate
- Critical engagement with primary and/or secondary sources

iv. Organization and presentation:

- Clarity and coherence of structure
- Clarity and fluency of writing, code, prose, or presentation (as relevant)
- Correctness of conformity to conventions (code, grammar, spelling, punctuation or similar relevant conventions)

b. US grades for courses are defined according to the following rubric:

97-100

Work will be so outstanding that it could not be better within the scope of the assignment. These grades will be used for work that shows exceptional excellence in the relevant domain; including (as relevant to the domain): remarkable sophistication and mastery, originality or creativity, persuasive and well-grounded new ideas or methods, or making unexpected connections or solutions to problems.

94-96

Work will excel against each of the General Criteria. In at least one area, the work will be merely highly competent.

90-93

Work will excel in more than one area, and be at least highly competent in other respects. It must be excellent and contain: a combination of sophisticated engagement with the issues; analytical precision and independence of solution; go beyond paraphrasing or boilerplate code techniques; demonstrating quality of awareness and analysis of both first principles or primary evidence and scholarly debate or practical tradeoffs; and clarity and coherence of presentation. Truly outstanding work measured against some of these criteria may compensate for mere high competence against others.

87-89

Work will be at least very highly competent across the board, and excel in at least one group of the General Criteria. Relative weaknesses in some areas may be compensated by conspicuous strengths in others.

84-86

Work will demonstrate considerable competence across the General Criteria. They must exhibit some essential features addressing the issue directly and relevantly across a good range of aspects; offer a coherent solution or argument involving (where relevant) consideration of alternative approaches; be substantiated with accurate use of resources (including if relevant, primary evidence) and contextualization in debate (if relevant); and be clearly presented. Nevertheless, additional strengths (for instance, the range of problems addressed, the sophistication of the arguments or solutions, or the use of first principles) may compensate for other weaknesses.

80-83

Work will be competent and should manifest the essential features described above, in that they must offer direct, coherent, substantiated and clear arguments; but they will do so with less range, depth, precision and perhaps clarity. Again, qualities of a higher order may compensate for some weaknesses.

77-79

Work will show solid competence in solving problems or providing analysis. But it will be marred by weakness under one or more criteria: failure to fully solve the problem or discuss the question directly; some irrelevant use of technologies or citing of information; factual error, or error in selection of technologies; narrowness in the scope of solution or range of issues addressed or evidence adduced; shortage of detailed evidence or engagement with the problem; poor organization or presentation, including incorrect conformity to convention or written formatting. They may be characterized by unsubstantiated assertion rather than argument, or by unresolved contradictions in the argument or solution.

74-76

Work will show evidence of some competence in solving problems or providing analysis. It will also be clearly marred by weakness in multiple General Criteria, including: failure to solve the problem or discuss the question directly; irrelevant use of technologies or citing of information; factual errors or multiple errors in selection of technologies; narrowness in the scope of solution or range of issues addressed or evidence adduced; shortage of detailed evidence or engagement with the problem; poor organization or presentation, including incorrect conformity to convention or written formatting. They may be characterized

by unsubstantiated assertion rather than argument, or by unresolved contradictions in the argument or solution.

70-73

Work will show evidence of competence in solving problems or providing analysis, but this evidence will be limited. It will be clearly marred by weakness in multiple General Criteria. It will still make substantive progress in addressing the primary task or question, but the work will lack a full solution or directly address the task; the work will contain irrelevant material; the work will show multiple errors of fact or judgment; and the work may fail to conform to conventions.

67-69

Work will fall down on a number of criteria, but will exhibit some of the qualities required, such as the ability to grasp the purpose of the assignment, to deploy substantive information or solutions in an effort to complete the assignment; or to offer some coherent analysis or work towards the assignment. Such qualities will not be displayed at a high level, and may be marred by irrelevance, incoherence, error and poor organization and presentation.

64-66

Work will fall down on multiple General Criteria, but will exhibit some vestiges of the qualities required, such as the ability to see the point of the question, to deploy information, or to offer some coherent work. Such qualities will be substantially marred by irrelevance, incoherence, error and poor organization and presentation.

60-63

Work will display a modicum of knowledge or understanding of some points, but will display almost none of the higher qualities described in the criteria. They will be marred by high levels of factual or technology error and irrelevance, generalization or boilerplate code and lack of information, and poor organization and presentation.

0-60

Work will fail to exhibit any of the required qualities. Candidates who fail to observe rubrics and rules beyond what the grading schemes allow for may also be failed.

c. Synchronous Meeting Discussions and *Viva Voce* Examination Template

Synchronous meeting discussions and *viva voce* examinations are conducted on the same format: written work is submitted in advance, and a discussion follows. This provides students an opportunity to clarify and explain their written claims, and it also tests whether the work is a product of the student's own research or has been plagiarized.

For the *viva voce* examination, the submitted work is graded, and the grade is recorded prior to the oral examination.

The synchronous discussion and *viva voce* examination acts to shift the recorded grade on the submitted essay according to the following rubric:

+3

Up to three points are added for excellent performance; the student displays a high degree of competence across the range of questions, and excels in at least one group of criteria. Relative weaknesses in some areas may be compensated by conspicuous strengths in others.

+/- 0

The marked script is unchanged for fair performance. Answers to questions must show evidence of some solid competence in expounding evidence and analysis. But they will be marred by some weakness under one or more criteria: failure to discuss the question directly; appeal to irrelevant information; factual error; narrowness in the range of issues addressed or evidence adduced; shortage of detailed evidence; or poor organization and presentation, including consistently incorrect grammar. Answers may be characterized by unsubstantiated assertion rather than argument, or by unresolved contradictions in the argument.

-3 (up to three points)

Up to three are subtracted points for an inability to answer multiple basic questions about themes in the written work. Answers to questions will fall down on a number of criteria, but will exhibit some vestiges of the qualities required, such as the ability to see the point of the question, to deploy information, or to offer some coherent analysis towards an argument. Such qualities will not be displayed at a high level or consistently, and will be marred by irrelevance, incoherence, error and poor organization and presentation.

0

Written work and the oral examination will both be failed if the oral examination clearly demonstrates that the work was plagiarized. The student is unfamiliar with the arguments of the essay or the sources used for those arguments.

Curriculum

Android App Development

Course Description

Android App Development teaches students to develop applications for Android, the most-used mobile platform in the world. Students learn to build with Android Studio and Kotlin, the modern toolkit used for professional Android development. The course prepares students to become professional Android developers.

The course begins with the fundamentals of Kotlin, the official language for Android development. Students learn its basic syntax, as well its object-oriented programming principles and key language features such as classes, collections, higher-order functions, and extensions. Students then delve into building Android apps, first by exploring layout and how to use common UI components. They learn Activity Lifecycle, and how to monitor and handle app states as users navigate through an app. Students learn how to create dynamic applications that persist data, and how to use APIs to pass data. The second part of the

course covers advanced functionality including animations, notifications, offline caching, and authentication and with Firebase.

Students learn by building small programming projects. The course culminates in a final group project where students build an Android app of their choice from scratch, going from initial design to deployment in the Play Store. Students will have obtained enough knowledge and have the option to obtain the Associate Android Developer certification offered by Google.

Applied Computer Science

Course Description

This capstone course enables students to demonstrate their proficiency in the technical and human skills that they have acquired throughout their undergraduate studies. The capstone requires students to conceptualize, plan, and implement a software project to completion, and evaluate their project's processes and outcomes.

The capstone builds on the initial project scoping work that was carried out in Capstone Research Methods, which culminated in students submitting a project proposal, and gaining formal approval for their capstone Project Proposal.

In this course, students will implement their proposed project with the support of a supervisor. Students with a common supervisor will be put into capstone advisory peer groups and will be required to meet with their group and supervisor regularly to update each other on their capstone progress and to provide feedback. Students will also have regular meetings with their capstone supervisor to provide additional support and guidance throughout the course.

Upon completion of their capstone projects, all students will be required to participate in a capstone symposium at the end of the term, where they will present their working projects/prototypes to internal and external stakeholders.

Artificial Intelligence

Course Description

Artificial Intelligence (AI) aims to teach students the techniques for building computer systems that exhibit intelligent behavior. AI is one of the most consequential applications of computer science, and is helping to solve complex real-world problems, from self-driving cars to facial recognition. This course will teach students the theory and techniques of AI, so that they understand how AI methods work under a variety of conditions.

The course begins with an exploration of the historical development of AI, and helps students understand the key problems that are studied and the key ideas that have emerged from research. Then, students learn a set of methods that cover: problem solving, search

algorithms, knowledge representation and reasoning, natural language understanding, and computer vision. Throughout the course, as they apply technical methods, students will also examine pressing ethical concerns that are resulting from AI, including privacy and surveillance, transparency, bias, and more.

Course assignments will consist of short programming exercises and discussion-oriented readings. The course culminates in a final group project and accompanying paper that allows students to apply concepts to a problem of personal interest.

Back End Development

Course Description

Back End Development builds on previous knowledge of web development and security, and equips students with knowledge of server-side development so that they can become professional back-end developers and build enterprise-scale applications. Students learn to develop and deploy server-side applications with greater scope and complexity.

In this project-based course, students deepen their understanding by building the back end for a cross-platform application. The project will require implementing advanced features that add complexity and uniqueness to a server's structure. Examples of these include payment gateways, chat rooms, full text search, WebSockets, etc. Students will design and build out all of the API endpoints needed for the application and properly secure them for use in any web or mobile front-end application. In doing so, they will explore the differences and tradeoffs between web services, APIs, and microservices. They will learn best practices for code quality including unit testing and error handling. They will also learn to efficiently document their APIs.

Students will understand key Developer Operations (DevOps) practices including environment design, testing, development controls, and uptime management. They will implement modern DevOps workflows (e.g., containers, cloud virtual machines), and learn tradeoffs between different approaches. They will set up continuous integration and continuous delivery, and explore various strategies for automated testing and application monitoring.

Capstone Research Methods

Course Description

The Capstone Research Methods course supports students in developing critical research skills that are needed for the successful completion of their capstone project (Applied Computer Science).

The course provides students with an overview of the research process and types of capstone projects that they can undertake, and includes a detailed exploration of relevant quantitative and qualitative research methods.

Students will develop skills in data gathering and analysis, researching and writing an effective literature review, creating a research proposal, and managing ethical considerations with regards intellectual property rights and research with human subjects.

At the conclusion of the course, students will be required to submit their formal capstone project proposal which should include details of their project scope, research question, hypothesis, and project plan. Their proposal must receive a passing mark before they are allowed to undertake the capstone course in the final term of the degree program.

Challenge Studio 1

Course Description

Engineering for Development, Challenge Studio 1, and Challenge Studio 2 are 3 courses that help students investigate the role that technology can play in solving some of the world's most intractable social and economic development challenges.

In Challenge Studio 1, students will work in groups to design, develop, and test a solution to a development challenge of their choice. The focus of this course is to provide students with the tools and skills to create meaningful technology solutions (e.g. services, products) to a sustainable development problem. This course builds on the problem identification and analysis skills that were developed in Engineering for Impact, the product management skills that were developed in Product Management and Design, and the ethical engineering skills developed in Ethics in Tech.

At the end of Challenge Studio 1 students will submit a Minimum Viable Product (MVP) that is ready to go to market as their final project deliverable.

The course will utilize virtual studio time, where groups work together on the key incremental tasks that are required to allow them to successfully create their final project output. Studio time will be supported by lectures, seminars, and learning resources on useful skills such as human centered design, end user identification, requirements gathering, value creation, impact measurement, and creative thinking and innovation.

Challenge Studio 2

Course Description

Engineering for Development, Challenge Studio 1, and Challenge Studio 2 are 3 courses that help students investigate the role that technology can play in solving some of the world's most intractable social and economic development challenges.

Challenge Studio 2 builds on the final output from Challenge Studio 1, and supports students in creating a sustainable business model for the MVP that they developed in the previous course. This course is focused on putting the MVPs in the hands of real users, getting their feedback, and iterating and refining the product or service, while also developing a viable business model.

The course will utilize virtual studio time, where groups are able to work collaboratively on their MVPs, with the support of additional lectures, seminars, and learning resources on important topics such as product launch planning, user evaluation tools and frameworks, business canvas development, funding models, financial modeling and strategy, and pitching.

The course will culminate in a pitch showcase, where students are required to present their work to relevant stakeholders (e.g. industry leaders).

Collaborating for Impact

Course Description

Collaborating for Impact aims to support students in developing effective collaboration skills in the pursuit of collective impact and success. Few problems are solved by individuals working alone, therefore the ability to work effectively with others to achieve a common goal is a crucial professional skill.

The course will start by focusing on the social awareness and relationship management components of Emotional intelligence, which were briefly introduced in the pre-requisite course, Optimizing Your Learning. Students will then be introduced to a variety of collaboration and leadership theories and frameworks, and they will examine theories of team dynamics and dysfunction, and reflect on how these relate to their past experiences of collaboration in academic and personal settings.

The second part of this course will require students to put their communication and collaboration skills to practice by completing a group project that is designed to test their ability to work effectively as a group and deliver a high-quality output under time, resource, and information constraints.

Communicating for Success

Course Description

Communicating for Success supports students in developing communication skills that are essential for success in their personal and professional lives. The course will focus on close reading, written communication, verbal communication, and non-verbal communication skills. An emphasis will be placed on weekly submissions, and peer and instructor feedback, to allow students to practice and improve their skills.

Students will learn how to effectively read and analyze texts as a precursor to developing their own written communication skills. They will then practice crafting clear communications by learning about topics such as writing structure and organization, grammar, audience awareness, and the iterative writing process. Next, students move on to verbal communication, and will learn how to confidently and skillfully deliver effective oral presentations. Finally, students will learn about the impact of non-verbal communication on how their messages are received.

The course will culminate in a project that will require students to develop and implement a strategy for communicating a technical topic to a non-technical audience.

Computer Systems

Course Description

This course explores computing beyond software. Students will go a level deeper to better understand the hardware and see how computers are built and programmed. It is modeled on the popular, project based “Nand to Tetris” textbook, which walks learners through building a computer from scratch. It aims to help students become better programmers by teaching the concepts underlying all computer systems. The course integrates many of the topics covered in other computer science courses, including algorithms, computer architecture, operating systems, and software engineering.

Students will learn how to build a computer system using progressive steps. The course starts with a brief review of Boolean algebra, and an introduction to logic gates. Students design a set of elementary logic gates using a Hardware Description Language. They then build chips to perform arithmetic and logical operations and build the computer’s main memory unit. Subsequently, students learn to write low-level machine language, and build a CPU to create a fully functional computer system. Finally, students implement a virtual machine, compiler, and basic operating system. Projects are spread out evenly throughout the course, and are completed in pairs.

By the end of the course, students will develop a strong understanding of the relationships between the architecture of computers, and software that runs on them.

Data Structures and Algorithms 1

Course Description

This course teaches the fundamentals of data structures and introduces students to the implementation and analysis of algorithms, a critical and highly valued skill for professionals.

Students start by examining the basic linear data structures: linked lists, arrays, stacks, and queues. They learn how to build these structures from scratch, represent algorithms using pseudocode, and translate these into running programs. They apply these

algorithms to real-life applications to understand how to make complexity and performance tradeoffs. Students will also learn how to develop algorithms for sorting and searching, use iteration and recursion for repetition, and make tradeoffs between the approaches. They will learn to estimate the efficiency of algorithms, and practice writing and refining algorithms in Python.

This course emphasizes big-picture understanding and practical problem-solving in preparation for technical interviews and professional practice. Throughout the course, students will solve common practice problems, and participate in mock interview sessions. As part of their regular assignments, they will also deepen their understanding of these topics and practice technical communication by writing technical blog posts.

Data Structures and Algorithms 2

Course Description

This course builds on Data Structures & Algorithms 1. Students will explore non-linear data structures, and implement and analyze advanced algorithms.

The course begins with a brief review of basic data structures and algorithms. Students deepen their understanding of searching and sorting, with a focus on describing performance. They learn about advanced data structures including priority queues, hash tables and binary search trees. Students build on their knowledge of graph theory to implement graph algorithms, and explore topics like finding the shortest paths in graphs, and applications of algorithms in maps, social networks, and a host of real-life applications. Other key topics include: divide and conquer recursion, greedy algorithms, dynamic programming algorithms, NP completeness, and case studies in algorithm design.

The course emphasizes big-picture understanding and practical problem-solving in preparation for technical interviews and professional practice. Students will solve common algorithmic problems and participate in mock interview sessions. As part of their regular assignments, they will write technical blog posts to deepen their understanding of these topics and to practice technical communication.

Designing Your Future

Course Description

Designing Your Future is inspired by the Stanford University course Designing Your Life. In this course, final year students will use design thinking to reflect on their undergraduate studies, and to plan and prepare for their transition into full-time employment

The course begins by exploring some of the self-awareness topics (e.g. identity, self-image, mental models, and motivation) that were introduced during the Optimizing Your Learning course. Students will revisit artifacts that they created during that course and reflect on their personal and technical growth over the course of the degree program. This will allow

them to craft a compelling personal and professional narrative so that they can position themselves effectively in the job market. Students will also learn how to use a design thinking approach to explore options for their careers after graduation.

In the second half of the course, students will develop practical skills to support their transition into full-time employment. They will learn critical skills like networking, professionalism, virtual collaboration, etc. that will support them in their integration into the workplace, while also developing some practical life skills, such as financial planning and emotional wellbeing.

Discrete Math

Course Description

This course builds on Mathematical Thinking and provides the mathematical foundation needed for many fields of computer science, including data science, machine learning, and software engineering.

It focuses on core mathematical areas that are essential in the toolkit of every computer scientist: logic, combinatorics and probability, set theory, graph theory, and elementary number theory. Each topic is covered with a focus on applications in modern computer science. It begins with a unit on logic which builds on previous knowledge, with an emphasis on writing readable and precise code. Probability and combinatorics focuses on analysis of algorithms and reliability. There is an in-depth focus on graph theory, and students explore the numerous applications of graph theory in computer science (data mining, clustering, networking, etc.). Finally, the course introduces number theory, beginning with fundamental results such as the Euclidean Algorithm then applications in cryptography.

The course culminates in a final group project where students explore original mathematical sources, and describe the historical proof techniques of a discrete math topic.

Engineering for Development

Course Description

Engineering for Development, Challenge Studio 1, and Challenge Studio 2 are courses that help students investigate the role that technology can play in solving some of the world's most intractable social and economic development challenges.

In Engineering for Development, students will learn how to analyze the root causes of development challenges so that they are able to build effective technology solutions. The course aims to introduce students to selected global development challenges using the United Nations Sustainable Development Goals (SDGs) as the framework for selecting the areas of focus.

Each term, the course will focus on 1- 2 subject areas (e.g. Quality Education, Affordable and Clean Energy, Climate Action), which will serve as test cases for students to develop the skills required to effectively analyze and understand complex development issues. Students will examine the system level dynamics that are at the root of these challenges, and will also analyze and critique technology related solutions that have been developed to address these challenges.

Ethics for Tech

Course Description

This course examines the ethical questions that are emerging as a result of rapid technological change. It prepares students to become responsible technologists, and provides a basis for ethical decision-making in their professional work.

It focuses on the promise and potential ethical dilemmas that result from the latest developments of computer science. Topics addressed include ethical decision-making, privacy and confidentiality, safety, manipulation/deception, and the impact of computers on society.

Students are first introduced to a selection of classical ethical theories (e.g., Rights, Justice/Fairness, Utilitarian, etc.). These provide a vocabulary and framework for examining ethical issues in tech. These frameworks are applied to a selection of relevant ethical questions. Examples might include: should social media companies suppress the spread of fake news on their platforms? Is electronic privacy a right? Are there justifiable uses of state surveillance? Should AI be used to “predict” crime by police? What ethical codes should guide AI like self-driving cars? Students read case-studies addressing these questions, write positioning papers, and offer feedback on papers of their classmates. Each live class is a seminar, where students engage in facilitated discussion.

The class culminates in an ethical redesign project where students analyze an existing product that poses ethical questions, then propose specific changes (in software, hardware, UI design, processes, features, implementation) to reduce ethical risks posed by the product.

Front End Web Development

Course Description

Front End Web Development builds on previous knowledge of web development, and extends students’ familiarity with modern HTML, CSS, JavaScript, and Web APIs. Students learn to develop and deploy client-side web applications with greater scope and complexity. Complex frontend features require using HTML, CSS, and JavaScript together. Students will usually have taken Web Application Development (or similar course under advisement from their faculty) as a prerequisite for this course.

Students deepen their knowledge of the JavaScript language, covering in depth topics like scope and higher order functions. Students practice using modern build tools for package management, bundling, optimization, formatting and linting, and testing.

Throughout the course, students will solve practice exercises and build projects, culminating in a final project using a JavaScript framework to build a complex web application.

Students will continue to apply technical communication skills by writing technical specs, drafting architecture diagrams, and documenting APIs. They will extend their communication practice through technical blogging on topics like tool comparisons, architecture choices, benchmarks, and frontend web design. Students will grow in independence by reading documentation to learn about novel language and browser features.

Industry Experience 1

Course Description

Industry Experience is a form of experiential learning that enables students to apply their academic knowledge in a professional context. Students work to build software that meets the needs of a professional organization by completing either (1) an approved internship, or (2) a product studio.

During the internship, students work on tasks that meet the needs of the organization, guided by an on-site supervisor. Internships must entail significant, substantial computer science. In the studio, external clients (e.g., businesses, non-profits) sponsor a software development project completed by students. A typical end result is a prototype of or a fully functional software system ready for use by the clients. These projects are completed by teams of 4-6 students, who meet with the client weekly to share progress and get feedback.

Students complete online courses under the supervision of a faculty advisor. Pre-work includes instruction in communication, goal-setting, and professional development. During the industry experience, students submit bi-weekly written reflections on their personal goals, challenges, and, for the studio, team feedback. At the end of the term, students obtain written feedback from their organization supervisor. They also submit a final report which describes the problem statement, approaches/methods used, deliverables, and skills gained. Industry Experience culminates in a final presentation which is shared as a public blog post.

Industry Experience 2

Course Description

Industry Experience 2 provides a form of experiential learning that enables students to apply their academic knowledge in a professional context. Students work to build software

that meets the needs of a professional organization by completing either (1) an approved internship, or (2) a product studio.

During the internship, students work on tasks that meet the needs of the organization, guided by an on-site supervisor. Internships must entail significant, substantial computer science. In the studio, external clients (e.g., businesses, non-profits) sponsor a software development project completed by students. A typical end result is a prototype of or a fully functional software system ready for an end user. These projects are completed by teams of 4-6 students, who meet with the clients or other end users weekly to share progress and get feedback.

Students complete online courses under the supervision of a faculty advisor. Pre-work includes instruction in communication, goal-setting, and professional development. During the industry experience, students submit bi-weekly written reflections on their personal goals, challenges, and, for the studio, team feedback. At the end of the term, students obtain written feedback from their organization supervisor. They also submit a final report which describes the problem statement, approaches/methods used, deliverables, and skills gained. Industry Experience culminates in a final presentation which is shared as a public blog post.

Interaction Design

Course Description

This course introduces students to the principles of human-computer interaction (HCI). Students explore how humans process information (perception, memory, attention) in the context of designing and evaluating interfaces. This course complements programming coursework by helping students understand how to design more usable systems.

The course builds on previous knowledge of design thinking and expects students to apply the design thinking methodology as a starting point. The first part of the course focuses on designing for multiple platforms. Students create designs that solve a problem on multiple devices (e.g., web, mobile, wearables) and learn how to create a coherent design system as users move between devices. The second part of the course delves into design beyond visual user interfaces, and teaches students how to design for emerging technologies, for example, sensors, controls and ubiquitous computing. Throughout the course, students learn and apply a variety of evaluation methodologies used to measure the usability of design.

This is a project-based course and, in each part, students will work in a team to design, prototype and test a solution to a problem. Students will present their designs in class sessions, and practice giving and receiving meaningful critiques.

Introduction to Data Science

Course Description

Data science is applicable to a myriad of professions, and analyzing large amounts of data is a common application of computer science. This course empowers students to analyze data, and produce data-driven insights. It covers all areas needed to solve problems involving data, including preparation (collection and integration), presentation (information visualization), analysis (machine learning), and products (applications).

This course is a hybrid of a computing course focused on Python programming and algorithms, and a statistics course focusing on estimation and inference. It begins with acquiring and cleaning data from various sources including the web, APIs, and databases. Students then learn techniques for summarizing and exploring data with spreadsheets, SQL, R, and Python. They also learn to create data visualizations, and practice communication and storytelling with data. Finally, students are introduced to machine learning techniques of prediction and classification, which will prepare them for advanced study of data science.

Throughout the course, students will work with real datasets (e.g., economic data) and attempt to answer questions relevant to their lives. They will also probe the ethical questions surrounding privacy, data sharing, and algorithmic decision making. The course culminates in a project where students build and share a data application to answer a real-world question.

Introduction to Programming in Python

Course Description

This course is intended for students with little or no programming experience. It aims to help students develop an appreciation for programming as a problem solving tool and to provide a foundation in Python programming. Students will learn how to think algorithmically, solve problems efficiently, and prepare for further computer science studies.

The course begins with an introduction to programming constructs in a simplified visual environment. Students will learn to specify commands, and to construct complex programs from simple instructions. The next part of the course introduces a formal programming environment, and students learn the basic syntax and structures of Python. Topics covered include variables, expressions, conditional execution, functions, loops, and iterations. Throughout the course, students will be exposed to abstraction and will learn a systematic way of constructing solutions to problems. They will work on team projects to practice pair programming, code reviews, and other collaboration methods common to industry.

The course culminates in a final group project and presentation during which students demonstrate and reflect on their learning.

iOS App Development

Course Description

iOS App Development teaches students to build modern mobile applications using Apple's iOS development platform and tool chain. It prepares students to become professional iOS developers. Students learn the core principles of the Swift programming language and Apple's front-end frameworks for creating single and multi-page applications.

The course begins with an introduction to the Swift programming language and Xcode, Apple's development environment for iOS. Students learn the core object-oriented programming principles of Swift, its Model/View/Controller paradigm, and its supporting classes. They then learn to create user interfaces with UIKit, Apple's front-end framework.

The second part of the course teaches students to create dynamic iOS applications that pass information between views and objects, and respond to user events. Students learn how to incorporate networking into their mobile apps, and how to get and parse information from the internet through APIs. They also learn techniques for creating asynchronous apps, including industry-standard patterns and frameworks to persist data locally and synchronizing data between the local device and the cloud.

Students learn by building small programming projects. The course culminates in a final group project where students build an iOS app of their choice from scratch, going from initial design to deployment in TestFlight.

Machine Learning

Course Description

This course aims to teach students the theoretical and practical methods for solving problems using machine learning. Machine learning is one of the fastest-growing areas of computer science. Its applications are reshaping society, from consumer products (e.g., voice assistants and recommendations) to life-sciences (e.g., personalized medicine and tumor detection). This course will build on the Data Science introductory course, and help students understand how, why and when machine learning methods work.

Students will be introduced to major paradigms in machine learning, and gain working knowledge of supervised and unsupervised learning techniques. Students will learn to solve common problems such as regression, classification, clustering, matrix completion and pattern recognition. They will learn how to train and optimize neural networks. They will explore modern software libraries that enable programmers to develop machine learning systems. Students will use these libraries along with publicly available data sets to build models, then learn how to evaluate and verify the results. Throughout the course, as they apply technical methods, students will also examine the societal context of machine learning and considerations of transparency, bias and privacy.

Course assignments will consist of short programming exercises and peer discussions of ethical considerations. The course culminates in a final group project and accompanying paper that allows students to apply concepts to a problem of personal interest.

Mathematical Thinking

Course Description

This course helps students develop the ability to think logically and mathematically. It prepares students for more advanced courses in algorithms and discrete mathematics. An emphasis is placed on the ability to reason logically, and effectively communicate mathematical arguments.

The course begins with a brief review of number systems, and their relevance to digital computers. Students review the algebraic operations necessary to perform programming functions. In the unit on logic and proofs, students learn to identify, evaluate, and make convincing mathematical arguments. They are introduced to formal logic, and methods for determining the validity of an argument (truth tables, proofs, Venn Diagrams). Students learn to decompose problems using recursion and induction, and how these methods are used in real-world computational problems. The final unit is an introduction to counting and probability. Topics covered include principles of counting, permutations, combinations, random variables, and probability theory.

Throughout the course, students apply their knowledge by solving logic puzzles and creating programs in Python.

Network and Computer Security

Course Description

Network and Computer Security teaches students the principles and practices of security for software, systems, and networks. It aims to make students critical examiners and designers of secure systems. Students will learn the mathematical and theoretical underpinning of security systems, as well as practical skills to help them build, use, and manage secure systems.

The first part of the course is focused on applied cryptography. Students learn general cryptographic protocols and investigate real-world algorithms. The second part of the course covers software and system security, including access controls, trends in malicious code, and how to detect system vulnerabilities. There is a special focus on web security, and modern practices for building secure web architectures. The final section of the course focuses on network security and covers concepts of networking, threats, and intrusion protection.

Course projects will require students to think both as an attacker and as a defender, and write programs that examine security design. Students will also examine recent security and privacy breaches. Working in pairs, they'll conduct an in-depth investigation, and give a presentation to help classmates understand its technical underpinnings and social implications.

Optimizing Your Learning

Course Description

Optimizing Your Learning aims to transform incoming first year students into effective and empowered self-directed learners.

In the modern world, long-term academic, professional, and personal success is driven by the ability of individuals to take control of their learning. Therefore, this course helps students to develop the knowledge, skills, and mindsets necessary to take ownership of their learning and build their self-efficacy.

During the course, students will develop competence in skills that are most critical for effective self-directed and self-regulated learning (i.e. self-management, self-monitoring, and self-modification), while also learning how to use learning strategies to maximize their overall learning efficiency and efficacy. They will also utilize the Emotional Intelligence framework to explore their identity, self-image, motivation, and self-regulation skills, to support their development as self-directed learners.

The course culminates in the creation of a personal learning charter that will help guide students in their learning throughout their undergraduate studies, which can also be applied to their learning activities in other realms of their lives.

Product Management and Design

Course Description

This course teaches students to build products users want and love. It gives students a foundation in the tools and practices of modern product management and interaction design. Students will work in pairs to apply product development skills to real user challenges.

The course begins with a focus on user research. Students learn and apply the design thinking framework to product development. They learn to define user needs through user interviews and market analysis. They learn to translate user needs into product specifications, and define metrics to test product success. They learn to create and test design prototypes (wireframes, user journeys). The second part of the course focuses on UX/UI design. Students learn key concepts in UI/UX design including information hierarchy, and typography and color. Students will create high-fidelity UI mockups using industry-standard tools. They'll then conduct usability tests to gauge the effectiveness of their designs.

As students work in pairs, they will practice the complementary and collaborative roles of PMs and UX designers in early product development. They'll also practice giving design critiques to other teams, and responding to feedback on their designs. By the end of

the courses, each pair will have a user-tested, refined, and development-ready design for a web or mobile application.

Programming 1

Course Description

The course helps students develop an appreciation for programming as a problem-solving tool. It teaches students how to think algorithmically and solve problems efficiently, and serves as the foundation for further computer science studies.

Using a project-based approach, students will learn to manipulate variables, expressions, and statements in Python, and understand functions, loops, and iterations. Students will then dive deep into data structures such as strings, files, lists, dictionaries, tuples, etc. to write complex programs. Over the course of the term, students will learn and apply basic data structures and algorithmic thinking. Finally, the course will explore design and implementation of web apps in Python using the Flask framework.

Throughout the course, students will be exposed to abstraction and will learn a systematic way of constructing solutions to problems. They will work on team projects to practice pair programming, code reviews, and other collaboration methods common to industry. The course culminates in a final group project and presentation during which students demonstrate and reflect on their learning.

Programming 2

Course Description

This course expands on Programming 1, and deepens students' knowledge of Python with a focus on data access and management.

Previously introduced programming topics include data types, operators, variables, and control flows are reinforced, this time in the context of retrieving and manipulating data. Students learn to use Regular Expressions, a powerful tool for finding and extracting data from string and other data types. They are introduced to modern web protocols, and learn how to retrieve data from web services using Python and JSON, and how to access and parse data in XML. Students learn the basics of working with databases and the relationships between databases. They learn how to write queries in SQL, the foremost programming language for generating, manipulating, and retrieving information from a relational database.

Students work on small projects throughout the course. The final project challenges students to retrieve and visualize original data in Python.

Programming in Python

Course Description

This course is intended for students who have completed Programming 1. It aims to help students deepen their programming skills, concentrating on Python. Students will learn how to think algorithmically and solve problems efficiently.

Students will gain proficiency in specifying commands, and constructing complex programs from simple instructions. The course uses a formal programming environment, and students learn the syntax and structures of Python. Topics covered include variables, expressions, conditional execution, functions, loops, and iterations. Throughout the course, students will be exposed to abstraction and will gain a systematic way of constructing solutions to problems. They will work on team projects to practice pair programming, code reviews, and other collaboration methods common to industry.

The course culminates in a final group project and presentation during which students demonstrate and reflect on their learning.

Team Software Project

Course Description

In this course, students practice the skills necessary to work effectively on a professional software product team. By working in small teams to build a web application, they integrate the technical, communication, and collaboration skills built in previous courses.

Students build a multi-feature web application, either for a fictional client or an original idea of their own design. As they work together, they learn modern technical collaboration tools and practices. Topics covered include using version control for shared repository management, writing technical design documents, and conducting code reviews. They also practice project management skills by implementing the SCRUM framework, including sprint planning, reviews, and retrospectives. During each milestone, team members rotate taking on various roles including Scrum master, product owner, and technical lead. Throughout the course, students will also apply and refine the emotional intelligence, team development, and leadership frameworks previously learned. By the end of the course, students should understand and value the various roles within a software product development team, and be able to participate effectively on a product team.

There are no scheduled class sessions. Teams will submit their sprint retrospectives for feedback from peers and faculty. The course culminates in a showcase where students present their final project to their peers and external stakeholders.

Web Application Development

Course Description

This course builds on Web Foundations, and provides a comprehensive introduction to client and server-side development for the web.

In this project-based course, students will work independently to build a web application, and progressively apply new knowledge to their application. Students deepen their knowledge of HTML and learn advanced CSS, including how to use CSS variables and modern frameworks for motion and interaction. They learn about accessible web design, and how to create websites and apps that work well on mobile devices, and that support use of assistive technologies like screen readers.

Students will build the front-end of a web application using HTML, CSS and JavaScript then write a supporting back-end using either a JavaScript or Python framework. In doing so, they will demonstrate knowledge of the request-response structure, database management, and JSON-based APIs. Students will also apply technical communication skills by writing technical specs, drafting architecture diagrams, and documenting APIs.

Web Foundations

Course Description

This course provides a foundation in building for the web. It helps students understand how the internet works, examines the role of the internet in their lives, and teaches them the basics of web development. The course prepares students for the advanced course in Web Application Development.

The course begins with a brief history of the internet and network technologies. Students will learn about the physical underpinnings of the internet, barriers to connectivity, and efforts to expand access (e.g., undersea cable projects, satellite projects). They will also explore the challenges of internet security and privacy. Students will be encouraged to make these social explorations personal, and investigate the history, barriers, and opportunities for connectivity in their local regions. The course will also cover the building blocks of web application development. Students will master HTML, intermediate CSS, and basic concepts and syntax of JavaScript.

The course culminates in a “Knowledge Share” project during which students create a website to educate a non-technical audience on a key aspect of the internet or emerging technology.