

CSC-535 User interface analysis & design (Web programming) 3 credits

Programming is one of the most difficult branches of applied mathematics; the poorer mathematicians had better remain pure mathematicians.

– E.W. Dijkstra

Instructor: Dr. Serge Kruk
Email: kruk@american.edu
Office: SCAN 112
Office hours: Every day at Noon
Textbook: None are required but the following are references
Javascript: The definitive guide, by David Flanagan
Designing Interfaces, by Jenifer Tidwell
Online references: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>
<http://www.djangobook.com/>
<https://www.djangoproject.com/>
<https://developer.mozilla.org/en/JavaScript>
Times: T,Th 1845h-2000h

Goals:

The main objective of this class is to give the students the necessary tools to develop web applications and understand their theoretical underpinnings: event-driven programming, client-server architecture and human interface design. To do this, we will cover the common protocols, markup languages and client-side language, as well as two development frameworks, Django (for python) and GWT (for Java). The major stress will be on providing a well-designed user interface as well as good engineering practice in software development.

Knowledge areas

From the ACM Computer Science Curriculum, this course covers the following knowledge areas (approximate hours between parentheses):

- PF5. Event-driven programming (4)
- NC1. Net-centric computing (2)
- NC2. Communications and networking (5)
- NC3. Net security (2)
- NC4. The web as client-server computing (3)
- NC5. Building web apps (8)
- NC8. Multimedia technologies (2)
- PL6. Object-oriented programming (2)
- HC1. Foundations of Human-Computer interactions (2)
- HC2. Building a simple graphical user interface (2)
- HC5. Graphical user interface design (3)
- HC6. Graphical user interface programming (4)

Topics covered:

- HTTP (A basic overview of the protocol in relation to the OSI-model)
- HTML (A detailed description of the main elements of HTML5)
- CSS (A detailed description of the main elements of CSS3)

- Javascript
 - An introduction to the language as a general-purpose language
 - A description of the DOM and how to use Javascript to dynamically alter web pages, provide feedback to the user, validate fields, etc ...
 - JQuery (and other libraries) to simplify programming
 - AJAX or how to use Javascript asynchronously (Catch-phrase: WEB 2.0)
 - Google Closure or how one can build entire applications in Javascript
 - DART, CoffeScript and the future tools on the client-side
- Principles of interface design (The Good, the Bad and the Ugly)
- Framework: Django
 - Model, View, Controller paradigm
 - Object-relation mapper
 - URL Design and routing
 - Template system
 - Internationalization
- Framework: Google Web Toolkit (How to hide the client-server divide)

Attendance:

Attendance is **mandatory**, especially since less than a third of the class time will be devoted to lectures. It will be devoted to code review and active participation is required. That is, after some introductory lectures, a typical class will consist of someone presenting his/her code to the class and having the whole class review and critique the code. Evaluation will be of the presenters and the reviewers.

Computers and Software:

Students should come to class with their laptops and should have installed a good browser (Chrome or Firefox+Firebug), Python and eventually Django (for the second part of the semester). If they choose to, they can use Java and GWT for the later projects instead of Python and Django.

Grading:

Code reviews:	20%
Short projects (2 or 3):	40%
Final project:	50%

Every student must be a code presenter at least once during the semester. Commenters accumulate points for every (intelligent) comment or question they ask a presenter. You do not have to comment on every presentation, but on the majority of them to accumulate all the allocated points.

The short projects will be web applications that are in some stage of completion and that you must complete or extend. The grade will be a function of completeness and correctness of the code. The projects **must** be done in groups of two, using pair-programming techniques.

The final project is a web application that you must design, implement and present to the class. It will be developed and submitted in stages (interface drawing, mock-up, etc...)

Conversion to letter grade:

100-93	A	92-90	A-	89-87	B+	86-83	B	82-80	B-
79-77	C+	76-73	C	72-70	C-	69-60	D	59-0	F

Late Policy:

Late projects are accepted (unless a solution is posted, of course) but at a cost of 20% deduction per window of 24 hours after the due time. (So, if you submit 1 second after the due time, you get a deduction of 20% of whatever your grade is. After four days, you get zero.) Missed exams may not be made up. If an emergency forces you to miss an exam, permission to be excused must be sought from me in advance, if possible. If granted, the final exam score will replace the score of the missed exam (or the midterm score, if the final is missed).

Project submission:

All projects must be submitted on Blackboard. I will ignore any project submitted to me via email.

Academic Integrity:

Plagiarism and academic misconduct are defined in the University Academic Integrity Code www.american.edu/academics/integrity. You should be familiar with what constitutes academic dishonesty. Any information taken from the internet, books, or anywhere else for use on your assignments must be cited. You are permitted to discuss your work with other students at the conceptual level only (that means not in front of a computer and not while taking notes). Your work must be entirely your own work (or your group's work if working in a group). Unless otherwise stated, all exams will be closed-book, closed-notes. Instances of plagiarism may be reported and could result in disciplinary action.