CMSI 370
Interaction Design

http://dondi.lmu.build/fall2019/cmsi370

Fall 2019—Doolan 222
TR 9:55–11:10am (01), 1:15–2:30pm (02); 3 semester hours
Office Hours: TR 11:15am–12pm, WR 4:15–6:30pm, or by appointment

John David N. Dionisio, Ph.D.
dondi@lmu.edu
Doolan 102; (310) 338-5782

Objectives and Outcomes

This course explores the computer science subfield of *interaction design* (IxD), a.k.a. *computer-human* (or *human-computer*) *interaction* (CHI/HCI). IxD seeks to understand human behavior when interacting with computing systems and studies metrics, techniques, and theories for achieving effective interaction. Long after you finish this course, my hope is that you will be able to:

1. **Appreciate and express the art and science of interaction design, including its theories, principles, methodologies, and role in software design and development.**
2. **Communicate, evaluate, and expand upon a user interface of your own design.**
3. **Demonstrate the fundamentals behind designing and implementing user interfaces.**

In addition to the course-specific content, you are also expected to:

4. **Follow disciplinary best practices throughout the course.**

Prerequisites/Prior Background

Intermediate to advanced proficiency in computer programming is *very* helpful, almost essential. Concurrent or prior taking of CMSI 386 Programming Languages provides exposure to common language concepts with varying syntax.

Materials and Texts

- Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 5th Edition, Addison Wesley/Pearson, 2009.
- Jakob Nielsen. *Usability Engineering*, Morgan Kaufmann, 1994. *(available online—see Brightspace)*
- Donald A. Norman. *The Design of Everyday Things*, Basic Books, 2002.
- Assorted handouts, articles, and sample code to be distributed throughout the semester.

The following text, especially Chapters 6–8, can serve as a programming tutorial and reference:

- John David N. Dionisio and Ray Toal. *Programming with JavaScript: Algorithms and Applications for Desktop and Mobile Browsers*, Jones & Bartlett Learning, 2011.

(this text uses pre-ES6 syntax: concepts still hold but the actual code we see will likely look different)

Course Work and Grading

Your final grade will be based on the percentage of the points you get for the following deliverables against the total number of possible points. "W" indicates a written assignment and "P" indicates programming work:

GitHub and YouTube account listing	n/a	20 points
Front-end design document	W	100
API setup/tutorial document	W	80
Front-end development	P	100
API integration	P	80
Front-end evaluation and design vision document	W	100
Reusable direct manipulation user interface component	P	100
Total		580 points

Percentages ≥ 90% get an A– or better; ≥ 80% get a B– or better; ≥ 70% get a C– or better. I may nudge grades upward based on qualitative considerations such as degree of difficulty, effort, class participation, time constraints, and overall attitude throughout the course.

Term Portfolio

Your accumulated writings and software for the semester comprise the *term portfolio*—the final, definitive artifact that demonstrates the course's outcomes. It is how you show that you have, indeed, accomplished the objectives of this course.

An assignment's number is its due date in *mmdd* format, and it is always due by 11:59:59.999pm of that date. Point values are based on the state of your assignments at that moment.

Formal Written Work

Your portfolio will include these pieces of formal written work:

- A design document for this front end, which you will envision, describe, and sketch out
- A setup/tutorial document for one or more web service APIs upon whose functionality you plan to build a front end application
- A culminating document that evaluates your application and envisions future enhancements

Demonstrate outcomes *1a*, *1b*, *2a*, *2b*, and *4d–4f* to maximize the points for these assignments.

Programming Work

The second major type of work in your portfolio is software. Programming work includes:

- Development of the envisioned front end app
- Integration with web service API(s)
- Implementation of a reusable direct manipulation user interface component

Demonstrate outcomes *3a*, *3b*, and *4a–4f* to maximize the points for these assignments.

Version Control & Automated QA

Version control is an indispensable part of today's computer science landscape in industry, the academe, and the open source community. We use version control heavily in this course: make sure that you get the hang of it.

None of the assignments can be completed (well) overnight; they should be the result of steady progress from the moment they are assigned to the date they are due. "One and done" submissions will negatively affect the final score.

For the programming assignments, as an incentive for making steady progress early and often, an automated QA (quality assurance) system will be connected to your assignment repository via *continuous integration* (CI). CI monitors your assignment repository and can trigger assorted automated activities when new work arrives. For this course, CI performs automated QA on your source code.

Submit your work-in-progress to have the system identify areas of improvement. These areas will appear in a QA report that will be linked to your commits. Eliminate these errors and warnings and make sure your programs work correctly "out of the box" by the due date to maximize your points.

I reserve the right to discontinue further evaluation of your work if the severity or amount of QA flags makes the process too difficult or disruptive.

Workload Expectations

In line with LMU's *Credit Hour Policy*, the workload expectation for this course is that for every one (1) hour of classroom instruction (50 scheduled minutes), you will complete at least two (2) hours of out-of-class work each week. This is a 3-unit course with 3 hours of instruction per week, so you are expected to complete $3 \times 2 = 6$ hours of weekly work outside of class.

Attendance

Attendance at all sessions is expected, but not absolutely required. If you must miss class, it is your responsibility to notify me about this and keep up with the course. The last day to add or drop a class without a grade of W is August 30. The withdrawal or credit/no-credit deadline is November 1.

Academic Honesty

Academic dishonesty will be treated as an extremely serious matter, with serious consequences that can range from receiving no credit to expulsion. It is *never* permissible to turn in work that has been copied from another student or copied from any source (including the Internet) without properly acknowledging that source. It is your responsibility to make sure that your work meets the standard of academic honesty set forth in:

http://academics.lmu.edu/honesty

Americans with Disabilities Act

Students with special needs as addressed by the Americans with Disabilities Act who need reasonable modifications, special assistance, or accommodations in this course should promptly direct their request to the Disability Support Services Office (DSS). Any student who currently has a documented disability (physical, learning, or psychological) needing academic accommodations should contact DSS (Daum 224, x84535) as early in the semester as possible. All discussions will remain confidential. Please visit *http://www.lmu.edu/dss* for additional information.

Topics and Important Dates

Correlated outcomes are shown for each topic. Specifics may change as the course progresses. University dates (italicized) are less likely to change.

August/ September	Introduction to the course; background and history of interaction design *(1a, 2a)*
August 30	*Last day to add or drop a class without a grade of W*
September 2	*Labor Day*
	Usability metrics *(1b, 2a, 2b)*; guidelines, principles, and theories *(1b, 2a, 2b)*; introduction to modern front end applications *(3a, 4a–4e)*
October	Overview of interaction styles *(1b, 2a)*; menus, forms, and dialogs *(1b, 2a, 2b)*; implementation of the menus, forms, and dialogs interaction style *(3a, 3b, 4a–4e)*
November	Direct manipulation *(1b, 2a, 2b)*; affordances *(1a, 1b, 2a, 2b)*; implementation of the direct manipulation interaction style *(3a, 3b, 4a–4e)*
November 1	*Withdraw/credit/no-credit deadline*
November 27–29	*Thanksgiving; no class*
December	Code review/improvement workshops *(1a–4e)*; miscellaneous topics *(varies; time permitting)*
December 13	*Last set of term portfolio assignments due*

You can view my class calendar and office hour schedule in any iCalendar-savvy client. Its subscription link can be found on the course web site (it's too long to provide in writing).

Tentative Nature of the Syllabus

If necessary, this syllabus and its contents are subject to revision; students are responsible for any changes or modifications distributed in class or posted to the course website.

Course Evaluations

Student feedback on this course provides valuable information for continued improvement. All students are expected to fairly and thoughtfully complete a course evaluation for this course. This semester, all course evaluations for the Seaver College of Science and Engineering will be administered online through the Blue™ evaluation system. You will receive an email notification at your Lion email address when the evaluation form is available. You may also access the evaluation form on the Brightspace dashboard during the evaluation period.

A few minutes of class time will be reserved for you to complete a course evaluation within a fortnight before finals week. Please bring a laptop, smart phone, tablet or other mobile device to class on this date so that you can access the online evaluation platform.

Course Outcomes

1	**Appreciate and express the art and science of interaction design, including its theories, principles, methodologies, and role in software design and development.**	
1a	*Understand and express how interaction design relates to mental models.*	This is derived mainly from Don Norman's big picture view of interaction design, as explained in *The Design of Everyday Things*.
1b	*Understand and describe core interaction design concepts: usability metrics; interaction design guidelines, principles, & theories; interaction styles; and affordances & natural mappings.*	For these outcomes, "understand and describe" includes being able to list, define, explain, and give examples of relevant concepts, always with clarity, coherence, intellectual force, and stylistic control.
2	**Communicate, evaluate, and expand upon a user interface of your own design.**	
2a	*Effectively use: usability metrics; interaction design guidelines, principles, & theories; interaction styles; and affordances & natural mappings to make appropriate, well-founded interaction design decisions, then communicate them in written form.*	Such decisions include user interface analysis, diagnosis of interaction design problems, evaluation or comparison of user interfaces, choosing interaction styles, and envisioning new user interface designs. Such choices or decisions must also be clearly explained or justified.
2b	*Assess the performance of a particular user interface, including but not limited to capturing and prioritizing usability metrics and correlating results to mental models and interaction design theories.*	Interaction design is not an entirely subjective endeavor, and can in fact be evaluated objectively and quantitatively. It is important to be able to conduct such evaluation, particularly on one's own designs, in order to map out genuine improvements to an application's usability.
3	**Demonstrate the fundamentals behind designing and implementing user interfaces.**	
3a	*Know and understand how user interfaces are constructed, especially the model-view-controller (MVC) paradigm.*	These outcomes are all demonstrated by writing programs that involve one or more of these areas. Thus, some specific set of technologies, languages, and libraries must be learned and used. However, it must also be understood that these concepts are general and technology-independent: when called for, one should be able to transfer this knowledge to other platforms.
3b	*Know and understand event-driven programming.*	
4	**Follow disciplinary best practices throughout the course.**	
4a	*Write syntactically correct, functional code.*	Code has to compile. Code has to work. No errors, no bugs. Use unit tests as much as possible.
4b	*Demonstrate proper separation of concerns, especially MVC.*	This is the basis of good software design. It makes software easier to maintain, improve, and extend. Proper separation of concerns includes but is not limited to correct scoping of variables & functions and zero duplication of code.
4c	*Write code that is easily understood by programmers other than yourself.*	This outcome involves all aspects of code readability and clarity for human beings, including but not limited to documentation & comments, spacing & indentation, proper naming, and adherence to conventions or standards.
4d	*Use available resources and documentation to find required information.*	The need to look things up never goes away. Remember also that the course instructor counts as an "available resource," so this outcome includes asking questions and using office hours.
4e	*Use version control effectively.*	In addition to simply using version control correctly, effective use also involves appropriate commit frequency and descriptive commit messages.
4f	*Meet all designated deadlines.*	