

CS290 – Web Development for CS Majors

Winter 2021, CRN: 33342, Credits: 4

Instructor: Joseph Jess

Web: <http://cf.linnbenton.edu/bcs/cs/jessj/web.cfm?pgID=5404>

email: jessj@linnbenton.edu

Initial class note: We are *not* in an introduction course to computer science (CS), though that does not mean most of the materials will be difficult. We will have an intro to various web development technologies and design ideas. We will assume that we have some programming experience and we will expand on the design, testing, and implementation concepts covered in an introductory CS course. We need to expect to do a good bit of reading, much practicing, and much discussion of the topics we cover.

1. LBCB catalog course description, including pre-requisites/co-requisites:

This course will cover how to design and implement a multi-tier application using Web technologies. This will include the creation of extensive custom client and server side code consistent with achieving a high-quality software architecture.

Prerequisites: CS162 Intro to Computer Science II with a grade of “C” or better or equivalent experience as determined by a Computer Systems Department instructor

2. Class Time-space:

2.1 Lecture + demo + lab: M 1000 – 1150, Internet, see Moodle for details

3. Measurable student learning outcomes:

At the completion of the course, students will be able to:

- 3.1 Describe the architectural elements of effective web applications.
- 3.2 Describe key threats to relevant architectural attributes in web enabled applications.
- 3.3 Demonstrate implementation of custom functionality across multiple tiers of a web enabled applications.
- 3.4 Evaluate which architectural strategies to apply to address quality requirements, with emphasis on scalability, usability and security.

My hopes are that we will also:

Design and create a moderately complex static web site that conforms to recent standards.

Implement a custom user interface behavior using client-side scripting.

Implement asynchronous calls for sending data between a client and a server.

Implement dynamically-generated websites using server-side scripting.

Use basic database commands to create, store and retrieve data in conjunction with a dynamic website

4. Learning resources:

4.1 **Note:** All class materials and storage will be freely available in a digital format

4.2 **HTML, CSS, Javascript, and select server-side languages** – available through several media and available options will be provided when needed in class.

4.3 (**recommended**) Access to a computer outside of class to practice and work on assignments.

4.4 A text editor. We could use a smart IDE to do this work as well!

4.4.1 We will discuss some capabilities of smart code editors during the course.

4.5 (**strongly recommended**) A desire to learn, experiment, design, test, and problem solve with code (both on and off of a computer).

5. Grading:

5.1 Scores for assignments will be available when the instructor gets to it... usually within the week of the due date. Grades will be made available through the Moodle gradebook.

5.2 Students will be required to turn in all coursework items before 23:59 (Pacific Time Zone) on the date that they are due (generally the first meeting day of the week in my courses)... though I have an extremely forgiving option for making up for missed work at the end of the term.

5.3 To earn a passing grade in this course you must pass each of the following coursework categories:

5.3.1 **Demonstration:** Discussion, quizzes, and weekly assignments – 50%

5.3.1.1 There are a number of discussion questions (usually turned in as a part of the programming projects), quizzes, and programming projects to be completed for this class, designed to challenge and solidify design, coding, and testing skills. Projects are generally graded based on the following rubric:

A. **Program Design (20%)**

Rating Criteria

- 20 Solution well thought out
- 10 Solution partially planned out
- 0 ad hoc solution; program was “designed at the keyboard” or no design submitted

B. **Program Execution (20%)**

Rating Criteria

- 20 Program runs very well under a variety of conditions, as submitted
- 10 Program runs much of the time, may be missing required files or instructions for libraries used
- 0 Program runs very poorly, not at all, or requires several modifications or files before it runs

C. **Specification Satisfaction (20%)**

Rating Criteria

- 20 Program satisfies specification completely and correctly
- 10 Important parts of the specification not implemented
- 0 Program poorly satisfies specification, or not at all

D. **Coding Style (20%)**

Rating Criteria

- 20 Well-formatted, understandable code and appropriate use of language capabilities
- 10 Code difficult to follow in one reading or poor use of language capabilities
- 0 Incomprehensible code, poor use of language capabilities, or a need to scroll up and down repeatedly

E. **Comments (20%)**

Rating Criteria

- 20 Concise, meaningful, and well-formatted comments and docstrings
 - 10 Partial, poorly written, or poorly formatted comments
 - 0 Wordy, unnecessary, incorrect, badly written or formatted, or none or nearly no comments
-

Note: careful design, systematic testing, consistent style, and readability of code are important software quality factors (all of which are subject to interpretation but graded by the instructor based on the spirit and letter of the requirements, so be sure to explain your decisions).

Note well: Your submission should be explained and able to be compiled and run from just your submitted files in your final submission for that project. This means that you need to include any files provided to you that are necessary for your project to compile or run.

Note very well: Source code and related documents submitted must be designed and implemented by the student submitting the work and any code must compile and run on one of the instructor's machines in order to be graded. (to create a working program quickly: get it working simply, then add to it; if at some point it stops compiling you will better know where an error was introduced)

5.3.2 **Final:** Final project – 50%

5.3.2.1 There will be a final project to test the overall ability to understand, design, implement, test, and reflect on the problem solving and programming knowledge and skills covered in the class.

5.3.2.2 The final project will be a mix of “in-class” (initial design, testing, and implementation discussion) and take-home (finalizing design, testing, and implementation) elements.

5.3.3 **Make-up Work:** I do not care when you learn it, as long as you learn it. To support this idea I allow missing work, even after the term for many, to be made up for in the final project (just be sure I know you are trying to make it up in the final project submission!).

5.3.4 Final grades will be given out based on the following based on score in the class:

- 90-100%: A
- 80-89%: B
- 70-79%: C
- 60-69%: D
- 00-59%: F

5.4 Reminder: A passing grade in order to count for course requirements for CS classes is generally a C or above.

6. **Other Administrative Information:**

6.1 For a list of general administration information (note that this list is not intended to be exhaustive), such as:

- 6.1.1 contacting me,
- 6.1.2 [accessibility resources](#),
- 6.1.3 expectations of student conduct,
- 6.1.4 communications,
- 6.1.5 student assistance,
- 6.1.6 miscellany,
- 6.1.7 nondiscrimination & nonharrasment,

(each section contains a number of sub-sections and is not meant to be exhaustive of all situations)

see my administrative information document: [administrative_information document](#).