Course Description

This course explores techniques that are used to design and implement web applications – both server side and client side code. Using open source (free) development tools such as Netbeans, Apache, Tomcat, and MySQL Workbench, students write code in the following languages:

- SQL (for the MySQL database server),
- Java/JSP (to create Web APIs for the Tomcat web application server), and
- HTML (content structure of web page within browser), CSS (styling of web page within browser) and JavaScript/ajax (code run by the browser, invokes Web APIs to update the User Interface within the browser).

Each student will select their own topic for their database and web application. Almost any topic can be selected, but there are some data model restrictions to keep projects (1) manageable and (2) aligned with the weekly homework assignments.

- Students create their own MySQL database and populate it with data.
- Students learn how to create a web application UI (User Interface) using HTML and CSS. To achieve user interface code reuse, students will write AJAX calls (a technique using JavaScript to make asynchronous HTTP requests).
- Students learn how to publish and test their web applications using internet protocols, such as, ftp (sftp) and http (https).
- To get data from their database, students create Web APIs (server side code) then write the client side code (JavaScript/AJAX) to invoke those Web APIs. Students will learn about JSON (a format typically used to transfer data over the internet).
- To update the database (insert, update, delete), students will write other Web APIs that accept and validate user data (using the server side request object to extract data from the user).
- To implement log on and security, students will employ the server side objects: session (to store user information such as logon status), and response (to redirect the user to an error page if they are not logged on).
- Student code will be written according to software design patterns such as MVC (Model-View-Controller), SRP (Single Responsibility Principle), and DRY (Don’t Repeat Yourself).

Although each student’s web application will employ HTML/CSS to provide some aesthetic appeal, web design is not the major thrust of this course. Instead, this course focuses on designing and implementing client side and server side code to create a reliable, secure, extensible, and maintainable web application. Rather than learning a specific web application development framework (which come and go rapidly), students learn about the building blocks upon which such frameworks are created. As part of their tutorial presentation, some students may share information about various web development frameworks with the class.
Syllabus for CIS 3308 Web Application Programming

Prerequisites

- Grade of C- or better in CIS 2107 Computer Systems and Low-Level Programming
- Grade of C- or better in CIS 2168 Data Structures

The only assumption is that students have knowledge and skills obtained in the pre-requisite courses listed just above. All other topics will be introduced as new material, even though some students may already have had some exposure. Students with more experience can add extra functionality to their weekly assignments (if they wish), as long as they meet all homework requirements and submit on time.

Textbook

There is no text book. Web references and other materials will be posted online.

Learning Objectives

By the end of this course, each student should be able to:

- Create a web application (client side and server side) using these languages: HTML, CSS, JavaScript, Java/JSP, and SQL.
- Use MySql Workbench (an open source graphical front end to MySql) to create a normalized database, populate it with data. Write SQL select statements that join database tables and write SQL statements that insert, update, and delete records.
- Be able to write an acceptable (look and feel) user interface that employs code reuse (using JavaScript/Ajax).
- Understand how database constraints (like primary key, foreign key) affect a web application. Write server side (Java/JSP) code that prevents database errors and/or appropriately handles database exceptions. Ensure that your code does not have any “database connection leaks”.
- Write Web APIs (server side java/JSP code) that interact with the database (insert/edit/delete/search).
- Write JavaScript (client side code) to invoke Web APIs (this technique is called AJAX).
- Understand the role and benefits of various “frameworks” such as jQuery (commonly used JavaScript library) and Bootstrap (commonly used CSS/JS framework that provides responsive design so that your layout looks good on a mobile device as well as on a desktop) and Angular / React (Single Page Application Frameworks).
- Understand and employ good programming practices that result in reliable and maintainable web applications, (e.g., software design pattern, such as Model-View-Controller, Single Responsibility Principal, and Don’t Repeat Yourself) as well as the proper handling of database connections.
Course Format

**Homeworks.** Almost every week, there will be a programming assignment that is highly related to (reinforcing) lecture topics. To get a grade for your homework you have to

- Complete the homework assignment and test it locally.
- Publish it and test what you published.
- Upload a zip file of your WHOLE web application (NetBeans project) into Canvas.
- Demonstrate homework functionality (from what you published) to the TA during lab period. There should be two TAs – one to help with and grade the lab activity and one to grade the homeworks.
- If you have not completed your homework by the due date, you can still complete it the following week (with a -20% penalty). After that, homeworks are not accepted, but you still have to complete the work to avoid further deductions on your project grade. (The project is just the culmination of all your labs, regression tested so that everything all works together.)
- If there is ever any question about a Homework grade, we will go by the code that you have uploaded into Canvas.

**Lab Activities.** Almost every week, during your lab period, there will be a graded lab activity.

- Lab activities are self-guided activities designed to show you how to get started on your homework for the week.
- These activities must be started and completed and graded during the lab period – there are no time extensions and your grade will reflect the milestones that you were able to complete. Even if you do not completely understand everything that you are doing in lab, try to keep a reasonable pace so that you can finish as many milestones as possible. Once you are working on your homework, you’ll have time to work at a slower pace to think about and explore what you are learning.
- In order to get a grade for the lab activity, you must demo your code to the TA and upload a zip file of your lab activity into Canvas -- before you leave lab. If there is ever any question about a Lab Activity grade, we will go by the code that you have uploaded into Canvas.
- There will be no Make-Up Lab Activities. If a student misses a lab activity (or arrives too late to complete the work) for any reason, their grade will reflect were able to complete. Exceptions: if a student has a long term documented illness, we may be able to make an accommodation. Ask your instructor if you encounter such a problem.

**Project.** The student’s project is the culmination of all the labs. If you have incorporated each homework into a single NetBeans project and performed regression testing each week (and followed the design specifications of each homework), you should not have a lot of work to do to submit your project. While the weekly homework grading may just focus on functional testing, the project grading includes a code review of the source code, so it’s best to keep your code clean and well designed as you go along. This includes things like having your code properly indented, appropriate use of white space/comments, no unused code, DRY (Don’t Repeat Yourself), SRP (Single Responsibility Principal), and MVC (Model View Controller). Typically the design specifications of each homework (if you follow them) should ensure that your code is well structured.

**Tutorial.** Each student will write and publish a tutorial on an approved topic. Selected students will present their tutorials to the class towards the end of the semester.

**Tests.** There will be two tests. I will try to give you a practice test before the first test. The last test (comprehensive) will be administered the last day of class. There will be no final exam.
Tentative (and approximate) Grade Weights

Supervised Assessments

<table>
<thead>
<tr>
<th>Activity</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab Activities, must be completed in lab (7 @ 7 points each)</td>
<td>20%</td>
</tr>
<tr>
<td>First Test (100 points)</td>
<td>40%</td>
</tr>
<tr>
<td>Second Test (100 points)</td>
<td>40%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>

Unsupervised Assessments

<table>
<thead>
<tr>
<th>Activity</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekly Programming Homeworks (8 @ 10 points each)</td>
<td>29%</td>
</tr>
<tr>
<td>Project (culmination of all Homeworks, code review, 100 points)</td>
<td>36%</td>
</tr>
<tr>
<td>Tutorial (including presentation to class, 100 points)</td>
<td>36%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>

Your Course Grade is the MINIMUM of the two above grades (Supervised vs. Unsupervised). Why?

- Students can get too much help doing their homework (or mindlessly use sample code without experimenting and/or learning). So, unsupervised assessments cannot be allowed too much weight.
- However, if unsupervised activities have little or no weight, many students will not do their homeworks, and consequently not learn ANYTHING in this class. Not only would this hurt individual students, but it would also compromise the learning of the whole class.
- In summary, students must do well in BOTH the supervised assessments AND the unsupervised assessments.

Grade Scale

<table>
<thead>
<tr>
<th>Range</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>93-100</td>
<td>A</td>
</tr>
<tr>
<td>90-92</td>
<td>A-</td>
</tr>
<tr>
<td>87-89</td>
<td>B+</td>
</tr>
<tr>
<td>83-86</td>
<td>B</td>
</tr>
<tr>
<td>80-82</td>
<td>B-</td>
</tr>
<tr>
<td>77-79</td>
<td>C+</td>
</tr>
<tr>
<td>73-76</td>
<td>C</td>
</tr>
<tr>
<td>70-72</td>
<td>C-</td>
</tr>
<tr>
<td>67-69</td>
<td>D+</td>
</tr>
<tr>
<td>63-66</td>
<td>D</td>
</tr>
<tr>
<td>60-62</td>
<td>D-</td>
</tr>
<tr>
<td>0-60</td>
<td>F</td>
</tr>
</tbody>
</table>
Tentative List of Homeworks – check online for latest information and due dates.

1. **Project Proposal** and Data Model (creating a MySql data model using MySqlWorkbench).
2. **Database** created and populated with data (using MySqlWorkBench to help you generate SQL statements that create/modify table designs then insert/edit/delete data).
4. **Data Display.** Writing Web APIs that send all the data from a given database table. Creating data display pages (that invoke the Web APIs using JS/AJAX). Connecting JSP code to java classes. Database access from java.
5. **Log On.** Writing Web APIs that handle log on/log off, storing user data into the JSP implicit session object. Writing JS/AJAX code that invokes the Web APIs. Redirecting users to error page if they try to access a page for which they are not authorized.
6. **Insert.** Writing Web APIs that can insert records into a database and provide server side validation error messages. Understanding how client side validation can be added on top of this.
7. **Update.** Writing Web APIs that can update records from a database reusing insert validation code.
8. **Delete.** Writing Web APIs that can delete records (you’ll need some server side validation, for example, cannot delete a record if another record is pointing to it).
9. (Time Permitting) **Search / Report Writer.** User interface that lets the user search for records with several search criteria. Implementing this code using Order N number of statements (where N is number of criteria).

**Miscellaneous**

- The **CIS department computer labs** are NOT open 24/7. Learn the lab hours and adjust your schedule accordingly. If you want to work from home, **set up your own development environment.** I have tutorials and instructions that can help you with this. Those who promptly set up their own web development environment do much better in this course than those who do not.
- **Attendance:** If you must miss lecture or lab, check Canvas to see what material was presented and ask your classmates about anything else that may have been discussed.
- **Communication:** Please contact me as soon as possible if you think you are running into difficulties. Ask me, or your lab instructor, or another student for help AS SOON AS POSSIBLE.
- **Disability Disclosure:** Any student who has a need for accommodation based on the impact of a disability should contact me privately to discuss the specific situation as soon as possible. Student must provide me with a note from the office of Disability Resources and Services (100 Ritter Annex, 215-204-1280).
- **Academic Honesty and Ethics:** Temple University and I expect you to observe the highest ethical standards. When working in the lab or on your project, you may consult others, but the work you submit must be your own. Never share your answers with others. Never accept answers from others. Unless otherwise directed, all quizzes are closed book, closed computer. All violations of academic honesty will be handled according to university policy.