SOEN 387-F, Fall 2018

Web-Based Enterprise Application Design

M/W, 11:45-13:00, H 920

Sep. 4^{th} till Dec. 3^{rd}

Stuart Thiel

stuart.thiel@concordia.ca
`http://www.encs.concordia.ca/~sthiel/soen387`
EV 11.411
Office Hours: W 13:30-14:30
(514) 848-2424 x7211
Tutorial FA: M, 8:45-9:45, H 903
Tutor: Tim Liao
Tutorial FB: F, 11:45-12:35, H 903
Tutor: Tim Liao

Barring Midterm, Final and Office Hour times, this will likely remain accurate.

Course Description: This course is about the architecture and technology used to develope Web-Based Enterprise Applications. Whereasa a Software Architecture course will give you a broad idea of architectural patterns, some of which you will see again in this course, SOEN 387 will focus on those patterns directly applicable to web application development.

In this course you will learn about the following techinologies:
Hypertext Transfer Protocol (HTTP), Web mark-up languages and encodings. Document Object Models (DOM). Client/server and layered architectures for Web-based Enterprise Applications (WEA). Presentation, Domain and Data Source design patterns. Client-side programming. Java servlets and Java Server Pages. Authentication, security and transaction processing.

Prerequisite(s): COMP 353 - Databases ; COMP 354 - Intro to Software Engineering or SOEN 341 - Software Process; SOEN 287 - Web Programming

Organization: The course is a 3-credit course with 2.5 hours of lectures scheduled every week. There is also one 1-hour tutorials every week. You are expected to allocate a considerable amount of time in studying the text and working on the individual programming project. A qualified tutor will provide you with help should you require it, but is mostly there to give you practical guidance for using the environment. A POD will be available for more direct programming support.
Credits: 3

textbf Text(s):

- Patterns of Enterprise Application Architecture
  Author(s): Martin Fowler

- Enterprise Application Design Patterns: Improved and Applied
  Author(s): Stuart Thiel
  http://soenea.htmlweb.com/soenea/browser/Documentation/trunk/Thesis/Thesis_Stuart.pdf?format=raw

Some topics are from external web references to be provided.

Course Objectives:

The objective of this course is to familiarize students with advanced Enterprise-Application Design Patterns and approaches. This course will focus on how web-based applications are designed and built on both the client and the server, making use of Java Servlets, JSPs, MySQL and XML.

Grade Distribution:

Project	40%
Midterm Exam	30%
Final Exam	30%

Letter Grade Distribution: Grades will be curved around a B, unless the class does abysmally as a whole, then it will be curved around a B-. Two standard deviations below the mean is the threshold to **fail**. Two standard deviations above is a guaranteed A+.

Course Policies:

- **General**
  - I hate writing code on exams. I hate marking it more. I cannot guarantee that there will be none, but I will sure try to minimize it.
  - Quizzes and exams are closed book, closed notes.
  - Exams may draw on lecture notes, book material, assignments, tutorials, class exercises or any reasonable expectation of expertise that could be acquired from looking up how to do any of the exercises/tutorials/assignments.
- **Assignments**
  - There is an individual project broken into 2 assignment submissions. The purpose of this project is to develop practical experience and competency with the development of a simple webapp throughout the course. A test suite will be provided for each assignment deliverable, and students are responsible for making sure that their code works with this test suite. Grading is done using the test suite, and is done automatically. If you did not make your application work with the provided test suite, you will get zero on that submission. Marking will be done in-person, with a marker in the lab. You have 3 minutes to show the marker that it passes the tests.
  - Submissions will still be through EAS, as I will also review your code to make sure you are using the required patterns! `https://fis.encs.concordia.ca/eas/`.
  - Submissions that are not implemented using the required patterns/framework will be penalized.
  - Late Submissions will be penalized by 50% for each day late.
  - Assignments are due by 11:59pm on the day they are listed as due.
- **Attendance and Absences**
  I do not care if you show up. It is your responsibility to know what was covered and what you should know. I will try to keep things light, lively and informative, so please show up ready to engage with your classmates and myself.

- **Evaluation**
  The majority of the submission evaluation will be based on running test suites against submitted project. Students will be provided with full or partial test suites to gauge the status of their own submission before submitting and to guide them. The exam evaluation will incorporate some general-knowledge testing, but will focus on evaluating application of design and architectural patterns consistent with addressing enterprise application development problems covered in class.

- **The Midterm**
  There is one midterm. The purpose of the exam is to test if you (a) have understood the material of the lecture/book; (b) worked on the assignment project and learned the right lessons from the experience; (c) can provide a complete solution to at least one thought-requiring design/architecture question. Treat the midterm as a wakeup call. There will be no second midterm or a make-up class. The midterm will consist of short-answer questions and multiple-choice/multiple-answer questions.

- **The Final** The final exam covers the whole curriculum. It is likely to be a mix of multiple-choice, short-answer and long-answer questions. The final has the same three evaluation aims as the midterms (see above).

- **Graduate Attributes**
  As per the template provided by Dr. Joey Paquet:

  > As part of both the Computer Science and Software Engineering program curriculum, the content of this course includes material and exercises related to the teaching and evaluation of graduate attributes. Graduate attributes are skills that have been identified by the Canadian Engineering Accreditation Board (CEAB) and the Canadian Information Processing Society (CIPS) as being central to the formation of Engineers, computer scientists and information technology professionals. As such, the accreditation criteria for the Software Engineering and Computer Science programs dictate that graduate attributes are taught and evaluated as part of the courses. This particular course aims at teaching and evaluating 3 graduate attributes. The following is a description of these attributes, along with a description of how these attributes will be incorporated in the course.

  The following attributes are considered and evaluated within the course:

  1. Design
  2. Use of Engineering Tools
  3. Knowledge Base for Engineering, and
  4. Problem Analysis

  As per the description of Graduate Attributes:

  > **Attribute 1: Knowledge-base:** Knowledge of Hypertext Transfer Protocol (HTTP), web mark-up languages and encodings. eXtensible Markup Language (XML). Client/server and layered architectures for Web-based Enterprise Applications (WEA). Presentation, Domain and Data Source design patterns. Client-side programming. Java servlets and Java Server Pages. Authentication, security and transaction processing.

Attribute 2: Problem analysis: Analyze different problems and derive the requirements, design and implementation constraints for the deployment of web-based enterprise applications.

Attribute 4: Design: Use of eXtensible Markup Language (XML), client/server and layered architectures for Web-based Enterprise Applications (WEA), presentation, domain and data source design patterns. Use of class diagrams and sequence diagrams to express system design

Attribute 5: Use of Engineering tools:: Choice and use of appropriate tools for the development of web-based applications, including client- and server-side application development languages and tools.

Attribute 1 will be evaluated directly through the multiple-choice/multiple-answer questions on the exams.

Attribute 2 will be evaluated indirectly through the automated evaluation of the assignments and directly through multiple-choice/multiple-answer questions on the exams.

Attribute 4 will be evaluated directly in terms of implementation and validation quality through the automated evaluation of the assignments. Architecture and design elements of attribute 4 will be evaluated directly on long-answer questions on the exam.

Attribute 5 will be evaluated indirectly through the assignments.

- **Rights and Responsibilities**

What is plagiarism?

The most common offense under the Academic Code of Conduct is plagiarism which the Code defines as "the presentation of the work of another person as one's own or without proper acknowledgement" (Article 16a).

This could be material copied word for word from books, journals, internet sites, professor's course notes, etc. It could be material that is paraphrased but closely resembles the original source. It could be the work of a fellow student, for example, an answer on a quiz, data for a lab report, a paper or assignment completed by another student. It might be a paper purchased through one of the many available sources. Plagiarism does not refer to words alone - it can also refer to copying images, graphs, tables, and ideas. "Presentation" is not limited to written work. It also includes oral presentations, computer assignments and artistic works. If you translate the work of another person into French or English and do not cite the source, this is also plagiarism. If you cite your own work without the correct citation, this too is plagiarism.

In simple words:

DO NOT COPY, PARAPHRASE OR TRANSLATE ANYTHING FROM ANYWHERE WITHOUT SAYING FROM WHERE YOU GOT IT! DON'T FORGET TO USE QUOTATION MARKS!

Source: `http://provost.concordia.ca/academicintegrity/plagiarism`, Sept. 2015

Tentative Course Outline:
The weekly coverage might change as it depends on the progress of the class. However, you must keep up with the reading assignments.

Plagiarism:

The most common offense under the Academic Code of Conduct is plagiarism, which the Code defines as "the presentation of the work of another person, in whatever form, as one's own or without proper acknowledgement" (Article 19a). [1]

It's not hard, don't cheat. Don't copy other people's code. Don't copy code off the Internet and pass it off as your own (or for this course, just don't copy code off the Internet).

Graduate Attributes [2]

As part of either the Computer Science or Software Engineering program curriculum, the content of this course includes material and exercises related to the teaching and evaluation of graduate attributes. Graduate attributes are skills that have been identified by the Canadian Engineering Accreditation Board (CEAB) and the Canadian Information Processing Society (CIPS) as being central to the formation of engineers, computer scientists and information technology professionals. As such, the accreditation criteria for the Software Engineering and Computer Science programs dictate that graduate attributes are taught and evaluated as part of the courses. The following is the list of graduate attributes covered in this course, along with a description of how these attributes are incorporated in the course.

- **A knowledge base for engineering**: Demonstrated competence in university level mathematics, natural sciences, engineering fundamentals, and specialized engineering knowledge appropriate to the program. Knowledge of abstract data types: stacks and queues, trees, priority queues, dictionaries. Data structures: arrays, linked lists, heaps, hash tables, search trees. Design and analysis of algorithms: asymptotic notation, recursive algorithms, searching and sorting, tree traversal, graph algorithms.
- **Problem analysis**: Ability to use appropriate knowledge and skills to identify, analyze, and solve complex engineering problems in order to reach substantiated conclusions. Analyze problems and determine their constraints in order to make a choice as to what data structures and algorithms to use for their implementation.
- **Design**: Ability to design solutions for complex, open-ended engineering problems and to design systems, components or processes that meet specified needs with appropriate attention to health and safety risks, applicable standards, and economic, environmental, cultural and societal considerations. Use and compose appropriate data structures and algorithms to solve a variety of problems.
- **Use of engineering tools**: Ability to create, select, apply, adapt, and extend appropriate techniques, resources, and modern engineering tools to a range of engineering activities, from simple to complex, with an understanding of the associated limitations. Make educated choices as to what data structures and algorithms to use to solve problems following their respective strengths and constraints.

Learning Objectives

- **Knowledge base**: Demonstrate competence in fundamentals of data structures and algorithms.

[1] `www.concordia.ca\students\academic-integrity\plagiarism.html`, May 4th, 2017

[2] As per the last official COMP 352 outline

- **Problem analysis**: Analyze and state model limitations and elements of uncertainty. Formulate and calculate qualitative and quantitative qualities of the problems inputs and outputs. Estimate computational complexity. Evaluate and pick the most appropriate approach based on relevant criteria.
- **Design**:
  - Critique/evaluate many possible diverse solutions and use techniques to evaluate different solutions with sound arguments related to the problems requirements and constraints. Demonstrate thinking outside the box to create innovative solutions.
  - Develop a system architecture adapted to the systems application context and its requirements and constraints. Development and specification of internal and external software interfaces at different modularity levels. Describe a solution that presents enough details for implementation.
  - Write code according to design. Validate implemented systems against system requirements, specifications and constraints, as well as interface specifications.
- **Use of Engineering tools**: Demonstrate appropriate operational use of tools (e.g. algorithms, abstract data types, data structures, asymptotic complexity analysis) for specific tasks in a laboratory environment.

Important Notes [3]

1. **One credit** represents, for an average student, a minimum of 45 hours of workload spread across the various academic activities (Source: Article 16.1.2 of the Undergraduate Calendar). For an average student, this suggests **a minimum of 135 hours of workload** for a 3-credit course, including the time spent in lectures, tutorials, laboratories, examinations, and personal and team work.
2. Assignments will consist of a theoretical and a programming part. Each student must **independently** and **separately** prepare and submit her/his assignment.
3. Criteria used in evaluation of assignments:
   - Correctness and Testing: the program should conform to the specification given in the assignment. This includes the proper handling of special cases and error conditions and the providing of correct results. The submitted test cases take into consideration special cases and error conditions.
   - Design: the program should be constructed from coherent, independent, and decoupled functions. A function should usually access only its own parameters and local variables.
   - Style: the program should be general-purpose and well-organized.
   - Documentation and Layout: The documentation should consist of a well-annotated program and clearly formatted output. Helpful identifiers and a clear layout are part of documentation. The documentation should include the description of your design and the algorithm implemented.
   - Efficiency: The program must implement the most appropriate method.

[3] As per the last official COMP 352 outline, modified slightly as we have no marker

- Program-User Interface: The program should be easy to use.

4. Programming assignments: For all programming components of your assignments, you need to use Java version 8. You will be using the same computing facilities and the same computer account you used in previous courses (e.g., COMP 249). If you do not have a computer account, you can obtain it from the help desk at H-960 or EV 07.182. This account will give you access to the laboratories. For more information on CSE Computer accounts please visit the website: http://www.encs.concordia.ca/helpdesk/access.html. If you have your own computer and prefer to use it, you may do so, but be aware that your programs must compile and run with Java 8 at the Concordia laboratories.

5. Submission format: All assignment-related submissions must be adequately archived in a ZIP file using your last name as file name. The submission must contain your name and student ID. Use your "official" name only no abbreviations or nick names; capitalize the usual last name. Inappropriate submissions will be heavily penalized. Only electronic submissions will be accepted. Students will have to submit their assignments using the EAS system. Assignments must be submitted to the correct folder for assignments. Assignments uploaded to an incorrect folder will not be marked and result in a zero mark. No resubmissions will be allowed. For the Java programming assignments you have to submit the complete source code and the compiled files, which must be executable without changes. If this is violated you will get a zero mark for these parts of the assignments.

Day	Content
Mon. 19^{th} Sep.	Intro to WEA, course scope, web basics refresher
Mon. 24^{th} Sep.	Servlets, context, servlet interface, http method interface, servlet config. JSP, EL and Taglibs
Wed. 26^{th} Sep.	Architectural Styles, Layed Architecture, 3-Layer approaches, MVC, good design **Read:** Fowler: Chapters 1-2, 14.1 **Read:** Thiel: Chapter 2
Wed. 3^{rd} Oct.	Database connections/pooling, Transaction Scripts, Data-Layer Patterns, Interacting with the DB, RDGs **Read:** Fowler: Chapters 4, 9.1, 10.2 **Read:** Thiel: Chapters 3.1–3.4
Wed. 10^{th} Oct.	Forward/Include to JSP, Presentation Views, Template Views (mention Transform Views and XSLT) and using this to generate JSON (or HTML) **Read:** Fowler: Ch. 05, Sections 5.1,5.2,5.3,5.4
Mon. 15^{th} Oct.	Domain Model Approaches, Domain Objects, Active Record, TDGs, Data Mappers Inheritance with Domain Objects (refresher from Dr. Constantinides lecture) **Read:** Fowler: Chapters 9.2, 10, 12.7, 12.8, 12.9, 12.10 **Read:** Thiel: Chapters 3.7
Wed. 17^{th} Oct	Concurrency, Lost Updates, Inconsistent Read **Read:** Fowler: Chapters 5
Mon. 22^{nd} Oct.	Optimistic and Pessimistic Concurrency Management, HTTP/Idempotency and the beginnings of REST **Read:** Fowler: Chapters 12.1, 14.2, 16.1 **Read:** Thiel: Chapters 3.6
Wed. 24^{th} Oct.	Implementing Concurrency Management, versions and ACID
Mon. 29^{th} Oct.	Application Layer, Page Controller, Front Controller, Front Command, Use Cases **Read:** Fowler: Chapters 14.2-5, 14.7 **Read:** Thiel: Chapters 3.8
Wed. 31^{st} Oct	Midterm
Mon. 5^{th} Nov.	Unit of Work and Dependant Mapping Lazy Loading and Identity Maps **Read:** Fowler: Chapters 11.1, 12.4 **Read:** Thiel: Chapters 3.9, 3.10
Wed. 7^{th} Nov	Refactoring I, Domain Objects, Lazy Loading with Lists Refactoring II, General Use Identity Maps, UoW Caution, Organizing Data Source Layer **Read:** Thiel: Chapters 4
Mon. 12^{th} Nov.	Referential Integrity, Tying up this refactoring **Read:** Thiel: Chapters 4.4–4.6
Wed. 14^{th} Nov	Leveraging a Framework, SoenEA as an example framework WebSequenceDiagrams: Login Example **Read:** Thiel: Chapters 5
Mon. 19^{th} Nov.	REST, Endpoints, HTTP, Idempotency, AJAX WebSequenceDiagrams: Some related Example
Wed. 21^{st} Nov	Client-side tools/architecture, Filters/Wrappers, REST/AJAX
Mon. 26^{th} Nov.	Simplifying Sequence Diagrams Sequence Diagram Example of Concurrency
Wed. 28^{th} Nov	Simplifying Class Diagrams
Mon. 3^{th} Dec.	Final Review
Thu. 6^{th} Dec.	Final Exam