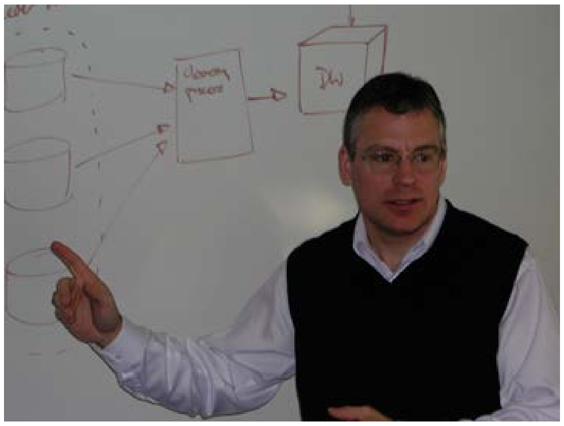
67-272: Application Design and Development

Carnegie Mellon University
Spring 2017 - Professor Heimann



This course provides students with the concepts and techniques to design and develop software applications, and to understand the design process. Students will learn the importance of user-centered design and will develop a prototype of a web application as a course project. In the process of developing the application, students will learn how to design and create relational databases, how to acquire competency in new programming languages quickly, how to use the Model-View-Controller pattern to develop software applications, how to ensure technical quality in software development, and how to apply principles of user-centered design. This course is a required professional core course and is open only to sophomores and juniors in the IS major who have completed 67-250 or equivalent.

It needs to be stressed to students that this course is not a programming class designed to make them fluent in one specific language. The goal of this course is to help students develop competency with several key technologies used in web development and e-commerce and to provide them with the principles needed to make effective use of these technologies. To further prepare students for a lifetime of learning, we will be teaching the actual programming segments of the class using the industry norm of "just-in-time" learning. We will give students an overview of the keys points of the programming language as needed for each upcoming project, but expect students to follow up on these presentations with reading and self-directed study to fill in any gaps in learning. Our goal is to enable students to be effective learners and compete in an industry where life-time learning is the norm.

Links at the top and bottom of the page should direct students to relevant course materials (schedule, course policies, project assignments and the like) that are being used for the Spring 2017 semester in Pittsburgh. The general policies link are for those adopted by the IS Program; all will be adhered to in this course.

Course Policies :: 67-272

Spring 2017

This page contains information on policies specific to the 67-272 course and applicable only for the Spring 2017 semester. For general Information Systems program policies, please choose the Department Policies link below. This section contains material on:

Grading
Labs
Questions & Piazza
Readings
Laptops
Concern for Students
Faculty
TA Information

Grades

Grades in this course are determined by student performance in four areas: course projects (5 phases, worth 40% total), three examinations (2 midterms and a final exam, worth 40% overall), weekly labs beginning in week one (worth 10% total) and attendance, short in-class assignments, and quizzes (periodically; worth 10% total). Any grading curves, if deemed necessary, will be applied only to the final course score and not to individual assignments.

Exams and the labs are designed to be completed by individuals without the assistance of classmates or other students. In this class, students may not look at the code of other students or show their code to other students. The IS department has made a separate statement regarding the honesty and integrity policy in this course and students need to review this policy as soon as possible. The IS program considers academic integrity to be of great importance, we actively scan for cheating policy violations and will take swift and appropriate measures against those who fail to abide by these standards.

We will have small in-class assignments to do on a periodic basis. These assignments will not be announced in class beforehand; since regular attendance is the norm, this should not be an issue. (FYI: attendance is taken until 5 minutes after class starts. After that time a student is considered absent. Students have two excused absences before any grade penalty is applied.) A major purpose of these in-class assignments is for both students and faculty to be certain that key concepts are understood and can be applied to basic problems. There will be no make-up for missed in-class assignments but you can be excused with prior permission.

Details on the course projects can be found on a separate page on the site. Being able to deliver work products on-time is important in the world of information systems, and for that reason, we will be firm on the deadlines associated with class assignments. In many cases late submission will not be allowed because a solution set will be released shortly after the assignment is due. In cases where late submission is allowed, any project turned in within 24 hours of the due date will receive an automatic 20 percent penalty. In all cases, assignments more than 24 hours late will not be accepted without a special exemption from Professor Heimann.

The dates for the project this semester will be:

- Phase 1: due January 26
- Phase 2: due February 19
- Phase 3: due March 19
- Phase 4: due April 9
- Phase 5: due May 5

Regrades: Any questions or concerns about grading must be directed to the Head TA responsible for grading for resolution before it can be taken to the professor. An entire statement regarding regrading for IS courses in general can be found in the department policies section and will be implemented here. **Do not labor under the mistaken impression that you somehow special and therefore are exempt from this policy!** If (and only if) you have followed the policy and are unhappy with the way the Head TA has handled your matter, you are welcome to take your case to the faculty. The faculty will want the Head TA's input before making a final decision (to be sure that we are fully informed when making the final decision), however, so **any attempt to bypass the Head TA will be futile**. If you attempt to do an end-run through the process, you will be sent back to discuss the matter with the Head TA. In the interest of fairness to all, we also reserve the right to lower grades further if we believe the TAs have been too generous in the grading the assignment in question. (This has happened in the past so be forewarned.)

Grading Course Projects: Programming projects will be graded based on their correctness, completeness, and their quality. We expect very high-quality work and attention to detail at all times in this course - it is up to you to see that this is so. We never want to see low-quality or mediocre work from any student. Carefully verify that your projects meet all stated specifications under a variety of test conditions and that they are eye appealing. Penalties will be assessed for errors, defects, and omissions based on their severity. Be aware that errors may cascade. Programs that are missing, substantially incomplete, do not load, do not run, or more than 24 hours late will be assessed penalties of 100%.

Labs

To help students master the material covered in class, we will have a series of hands-on lab assignments for students to complete, starting in week 1. These hands-on lab assignments must be started and completed during a scheduled lab session with either TAs or Prof. H. All labs are graded on a credit/no credit basis, depending on whether students completed the lab. (Come to lab each week, follow the instructions, and you get full credit -- very easy points to earn.)

Lab sessions are run by TAs Monday through Thursday evening in the Wean IS Lab (5336) from 7:30-9:30pm as well as one daytime lab on Wednesdays from 12pm-2pm in the Wean IS Lab (5336). Students must attend the lab section they are registered for, but each student has two 'switch coupons' that allow them to switch sections without penalty. To use your coupon, you must contact the Head TA for labs by Monday 6:00pm of the week you want to switch and notify him/her of your intentions. (This gives us time to restaff a lab if the expected attendance is higher than normal.)

Students are allowed one missed lab with no penalty (students who complete all labs get extra credit). Students are expected to be in lab on time; if students are tardy (arriving more than 5 minutes past the start time) for two labs, this will count as one absence. Each lab session is two hours long, but the labs themselves can be completed in less time, depending on the student. **No lab work done in advance will be accepted -- all work must be completed during the lab period.** If you show up with a completed lab you will receive a zero for the lab unless you delete the files in the presence of the TAs and begin again from scratch.

Questions & Piazza

Students are strongly encouraged to ask questions during class. The material can be tricky at times and we expect questions to be asked during lecture. Odds are that if you have a question, someone else is wondering the same thing; if no one asks then the mystery remains a mystery. In a few cases in the past, the question is on a more obscure technical point that interests very few in the class -- in those cases, Prof. H may choose to defer and answer the questioner after class so that the rest of the students are not bored or confused, but the question will be answered.

This term we will be using Piazza for class discussion and questions you might have outside of class. The system is highly catered to getting you help fast and efficiently from classmates, the TAs, and the professor. Rather than emailing questions to the teaching staff, we encourage you to post your questions on Piazza. The Head TAs will be monitoring Piazza every day and no question should go more than 24 hours without being answered (in most cases, much sooner). Do not send questions via email to the TAs without first checking Piazza to see if an answer has already been posted. In cases where the answer has already been posted. they simply tell you to go back to Piazza. If you email new questions that are not of a personal nature (like grades, standing in class) the TAs may ask you to post it on Piazza so they can answer it for everyone.

Find our Piazza class page at: https://piazza.com/cmu/spring2017/67272/home

If you have any problems or feedback for the developers of Piazza, please send an email to team@piazza.com

For the record, Professor Heimann is only half-Klingon. Hence, you can safely ask questions in class or during office hours without being worried about him 'killing you where you stand for asking your question.'

Readings

Below are listed two books assigned for this course, but I recognize that there are other books that teach the necessary concepts that could be used as a substitute. However, not getting any suitable references or delaying the purchase until much later in the semester represents a substantial risk and students should consider that decision carefully. I expect students to read the relevant sections of their texts in a timely manner. You can and will be tested on the reading -- even the parts I don't explicitly cover in class.

- Agile Web Development with Rails by S. Ruby et al.
- Designing the Obvious by R. Hoekman

Other books I recommend for this class are:

- The Well-Grounded Rubyist by D. Black
- Ruby Under a Microscope: An Illustrated Guide to Ruby Internals by P. Shaughnessy
- The Pragmatic Programmer: From Journeyman to Master by D. Thomas and A. Hunt

I have provided links to Amazon for each of these books, but there are multiple sources for each online and at a variety of price points. Students may purchase these books from any source they prefer.

Laptops

Because laptops, tablets, and cellphones can be a distraction to students in class (yourself and those around you) we will follow a policy similar to the MISM program and ban the use of laptops and similar electronic devices by students during class. There may be exceptions when you will need to do an exercise in class that requires a laptop -- you will be notified in advance if that is the case. Otherwise, please leave laptops in your book bags and turn all cell phones to silent mode prior to the start of class and leave them in your pockets or purses. (One exception: during exams, all cellphones must be turned off and left on top of your desk; cell phones cannot be taken out of the room during exams and will remain in the room during any restroom breaks.)

Concern for Students

All of us benefit from support during times of struggle. You are not alone. There are many helpful resources available on campus and an important part of the college experience is learning how to ask for help. Asking for support sooner rather than later is often helpful.

If you or anyone you know experiences any academic stress, difficult life events, or feelings like anxiety or depression, we strongly encourage you to seek support. Counseling and Psychological Services (CaPS) is here to help: call 412-268-2922 and visit their website at http://www.cmu.edu/counseling/. Consider reaching out to a friend, faculty or family member you trust for help getting connected to the support that can help.

Faculty

I maintain office hours that are run strictly on a first come, first served basis. However, I am available for appointments on other days and welcome students to stop in at other times without an appointment; if I am busy with something else at the moment then we will set up an appointment to talk at a more convenient time. Contact information is listed below:

- Professor Heimann
- Office: Porter 224A
- Phone: 8-8211
- Hours: Wed 10:30am-12:00pm & Tues/Thurs 3pm-5pm

Teaching Assistants

Below is a list of the four Head TAs this semester and their primary areas of responsibility:

- Mark Vella -- grading issues
- Ben Junker -- project issues
- Karen Irvine -- lab issues
- Israel Madueme -- general issues

If you have a concern, please talk with the TA primarily responsible for that concern. However, if you have difficulty getting ahold of that TA you may reach out to another TA. These TAs work as a team and any can help you in a pinch, but if the situation is not an emergency or urgent, do not be surprised if they refer you to the appropriate TA for that issue.

Lab TAs are primarily there to help with labs and answer questions on labs. They may be able to help with the project or other course related questions, but for the most part, those questions should either be put on piazza or asked of the appropriate Head TA.

The instructor reserves the right to make modifications to materials in this syllabus during the term as circumstances warrant.

Course Objectives for 67-272

Spring 2017

In this course, students will learn a new programming language and get additional experience working with relational databases. At the same time, this course is not primarily about learning a specific language or technology; rather it is about:

1. Understanding how to model data and translate those data models into working databases;
2. Becoming familiar with Model-View-Controller pattern in software architecture;
3. Better understand the object-oriented programming paradigm and how to use OOP concepts to create more coherent and stable software;
4. Understanding why and how to do software testing;
5. Learning how to use source code control to manage project development;
6. Knowing and applying principles of user-centered design to the development of software;
7. Utilizing all of the above to build web-based applications.

67-272 Lecture Schedule

Spring 2017

Below is a listing of the lectures and reading for each class. The schedule is only applicable for the current semester. This schedule is also tentative and may be changed as conditions merit it. If our pace is faster or slower than anticipated, then I will add or delete items at my discretion. Any changes will be announced in class and class announcements supersede the schedule presented here.

☐ **Lectures**

Date	Topic	Resources
January 17	Introduction; Git	☐ Agile Web Dev, ch. 1
January 19	Data Modeling & Use Cases	☐ Agile Web Dev, ch. 4
January 24	Ruby, Rails and MVC	☐ Agile Web Dev, ch. 3
January 26	Creating a Rails App	☐ Agile Web Dev, ch. 18 ☐ Proverbs example
January 31	Testing; Rails Models	☐ Agile Web Dev, ch. 7 ☐ PATS code example
February 2	More on Rails Models	☐ Agile Web Dev, ch. 19
February 7	Models and Business Logic	☐ Romanic Testing Code
February 9	Unit Testing	
February 14	Views & Controllers	☐ Agile Web Dev, ch. 20
February 16	Views & Controllers, Pt. 2	☐ Agile Web Dev, ch. 21
February 21	Exam Review	
February 23	Exam 1	
February 28	Review of Exam 1	
March 2	Ruby and Regex	☐ TWGR Regex Chapter
March 7	Designing Web Apps	☐ Designing Obvious, ch. 1-6
March 9	Designing Web Apps	☐ Designing Obvious, ch. 7-11 ☐ Article on color theory

March 21	Ruby and OOP	☐ Class examples
March 23	More OOP and Transactions	
March 28	Relationships & Rollbacks	☐ Arbeit code
March 30	Authentication & Authorization	
April 4	Exam Review	☐ Academic year code ☐ Bus route code
April 6	Exam 2	
April 11	Review of Exam 2	☐ Prime sets code
April 13	Design and Information	
April 18	Javascript in App Development	
April 25	Refactoring	
April 27	Design and Patterns of Behavior	
May 2	Deploying Web Apps	
May 4	Course Review	

About CMUIS :: Course Policies :: Course Objectives :: Department Policies :: Who Knows?

67-272 Labs

Spring 2017

☐ **Labs**

Due Date	Lab Title
January 21	Lab 1: Setup; Using Git
January 28	Lab 2: TempConverter; Using Foundation
February 4	Lab 3: SimpleQuotes
February 11	Lab 4: BookManager (part 1)
February 18	Lab 5: BookManager (part 2)
February 25	Lab 6: ChoreTracker
March 4	Lab 7: Views and Controllers
March 11	Lab 8: Writing Regular Expressions
March 24	Lab 9: Friends
March 31	Lab 10: Credit Cards
April 7	Lab 11: BandBlitz
April 15	Lab 12: My Town
April 28	Lab 13: Improving My Town
May 6	Lab 14: Pull Requests

67-272 :: Phase 1: Databases and Use Cases

Professor Heimann's sons (twins Alex and Mark) are looking to combine their love for chess with their entrepreneurial ways by starting up the A & M Chess Store, an online chess store that sells chess supplies to schools and other educational institutions at wholesale prices. Your job this semester is to construct an initial version of the Chess Store online application that will help schools order chess supplies and employees get those items shipped on time.

To help you in this endeavor, you are provided with several items that you will find links to at the bottom of this page. The first is a narrative describing important facts about the A & M Chess Store and what is needed from this online application. The remaining files are templates for documents that you will need in this phase.

In this phase we will focus on creating and documenting the data model that will be needed to create the online application. This phase will constitute 5 percent of your final course grade and is broken down into the following components:

1. Basic Data Model: Students will provide an entity-relationship diagram (computer generated; no hand drawings accepted), a database design in 3NF (see template below), and a data dictionary (see template below). All documents must be internally consistent.

2. Use Cases: Students will provide a set of uses cases that the system must implement to be considered fully working. Students will use the use case template provided below to structure their document.

Student solutions to this phase are to be printed out and turned in to the IS Office (Porter 222) by 4pm on January 26, 2017. Solutions will be handed out in class the next day, so **absolutely no late assignments will be accepted**.

If you have questions regarding the turn-in of this project or problems downloading any of the materials below, please post them on Piazza or ask someone well in advance of the turn-in date. Waiting until the day before it is due to get help is unwise -- you risk not getting a timely response that close to the deadline and will not be given an extension because of such an error.

Additional Information and Supporting Materials:

Case Narrative for A & M Chess Store
High-Level Use Case Template
Database Design Template
Data Dictionary Template
Use Cases Defined (from class)
Help files: Photos, inventory list, etc.

Due Date: January 26, 2017

Weight: 5.0

List of Projects

Phase 1: Databases and Use Cases

Phase 2: Models and Unit Testing

Phase 3: Basic Views and Controllers

Phase 4: More Models and Unit Testing

Phase 5: Finishing Chess Store

67-272 :: Phase 2: Models and Unit Testing

This phase we will continue our project to develop a foundation for the chess store system. In this phase and the next, we are focusing only on the inventory manager and his/her ability to adjust the inventory levels via purchases and to track the prices the chess store pays out for items over time. For now we will focus our efforts on building out the models for the entities specified in this phase's documentation and adding all necessary business logic. In addition, you will have to write unit tests for all models. This phase will constitute 7 percent of your final course grade and is broken down into the following three components:

1. **Creation of Models**: Models for the entities listed in the first ERD below and handed out in class need to be created. Below you are given a spec sheet to essential methods each model must have; those methods have to be fully implemented. The project must be **using Rails 4.2.5.x - 4.2.7.x** (not Rails 5).

2. **Unit Testing**: Unit tests for all methods in all models must be written and fully passing. We will check to make sure there is 100 percent code coverage for unit tests using the simplecov gem used in class and in lab. Because simplecov does suffer from some false positives (as mentioned in class) we may spot check tests to make sure certain tests were included. Only the models in this phase need complete coverage. There are steep penalties for less than 100 percent coverage and no credit at all for less than 90 percent coverage.

3. **Coding Style**: All code must be appropriately organized. What that means at this stage is the following: Related or similar functionality is grouped or clustered together. Indentation should be used consistently to make code readable. Comments showing organization should be present and explaining difficult code should be used when/if necessary.

Your project will be turned in via git -- at 11:59pm on Sunday, February 19, 2017, we will get a copy of your files in your remote git repository on darkknight.hss.cmu.edu. We will publish instructions on how to access your remote repository, but you should try to access your remote repository early in this phase; if you have not posted any code to the remote repository prior to February 14 and run into access issues in the last few days of the project there will not be much grace given and you can expect a stiff grade penalty. The solution for this phase will be released on the next day -- no late assignments will be accepted after solutions are released.

Again, if you have questions regarding the turn-in of this project or problems downloading any of the materials below, please post them on Piazza or ask someone well in advance of the turn-in date. Waiting until the day before it is due to get help is unwise -- you risk not getting a timely response that close to the deadline and will not be given an extension because of such an error.

Additional Information and Supporting Materials:

Model Specifications for Phase 2

List of Projects

Phase 1: Databases and Use Cases

Phase 2: Models and Unit Testing

Phase 3: Basic Views and Controllers

Phase 4: More Models and Unit Testing

Phase 5: Finishing Chess Store

Database Design for Phase 2
Data Dictionary for Phase 2
Instructions for Accessing Remote Repositories

Due Date: February 19, 2017

Weight: 7.0

67-272 :: Phase 3: Basic Views and Controllers

Overview

This phase we will continue our project to develop a foundation for the Chess Store system. In this phase we will focus our efforts on building an initial interface for chess store. This phase will constitute 6 percent of your final course grade and is broken down into the following four components:

1. **Implementation of Site Design**: A basic design for the site using CSS and appropriate images should be created. Unless students have strong CSS skills and already have a series of stylesheets they like to work with, students should use a CSS framework like Foundation or Materialize to build their application. At the same time, it is important to customize the framework for your own purposes; expect a grade penalty if you pretty much stick with all the defaults the framework provides. A home page for the application needs to be set up with some appropriate content, but design decisions about the exact contents are not being graded this phase. The student's name must appear on the bottom of every page in the footer (name should be listed as webmaster or designer). The information layout for customers, addresses and orders views as well as add/edit forms for these models should be appropriate.

2. **CRUD Functionality Established**: In this phase you will continue to use phase 2 data model, which is a simplified system. In addition, a list of high level uses cases is provided below to assist. A web based interface for doing all CRUD operations for the entities must be up and working. The cucumber tests provided must all pass (levels 0-4). To a large part this section will be assessed using tests discussed below and carries the bulk of the grade in this phase.

3. **Unit and Functional Testing**: All models along with all unit and functional tests (given as a solution to the previous phase) must be fully working. Since you were given a solution, expect a heavy penalty if this solution is not implemented and any unit tests fail. You are free to add new methods to the models, but be sure they are fully tested so your coverage does not fall below 100 percent.

4. **Coding Style**: All code must be appropriately organized. What that means at this stage is the following: Indentation should be used consistently to make code readable. Comments showing organization should be present and explaining difficult code should be used when/if necessary. All business logic should be in models, not views or controllers.

Starting the project

Starter code is available for you on Github. It is recommended that you begin with this code for two reasons. First, it has a complete set of models that are fully-tested and known to work for the task at hand. The second is that the

List of Projects

Phase 1: Databases and Use Cases

Phase 2: Models and Unit Testing

Phase 3: Basic Views and Controllers

Phase 4: More Models and Unit Testing

Phase 5: Finishing Chess Store

code contains a set of cucumber tests that will be used to assess a large portion of the app's CRUD functionality.

To get this code, it would be best to clone the existing repository with the following command:

```
git clone git@github.com:profh/ChessStore_Phase3_Starter.git
```

After this, switch into the project directory and remove the reference to `origin` with the following:

```
git remote rm origin
```

This will stop you from accidentally trying to push changes to Prof. H's repo (which won't be accepted). Now it is recommended that instead you set up a connection to your remote repository on darkknight soon and begin saving your code there early in your development. There is no limit on the amount of times you can commit code to the remote repository and remember that commiting to your local repository does not automatically mean the remote repository has been updated.

Running Cucumber tests

In this phase we will be using cucumber tests to verify that much of the basic CRUD functionality is in place. These are not comprehensive, but should cover a lot of the essentials and, along with controller tests, will count for a large portion of the grade in this phase (75%). The following command will run these tests:

```
bundle exec rake cucumber
```

Note that you will need to run `rake db:test:prepare` before your first cucumber test. If your design is very different and not able to pass the cucumber tests, discuss it with Prof. H to see if you are eligible for an exception. (All exceptions to cucumber tests must be granted in advance of the project's due date.)

Finishing the project

Your project will be turned in via git -- at 11:59pm on March 19, 2017 we will get a copy of your files in your remote git repository on darkknight.hss.cmu.edu. We have published instructions on how to access your remote repository, but you should try to access your remote repository early in this phase; if you have not posted any code to the remote repository prior to spring break and run into access issues in the last few days of the project there will not be much grace given and you can expect a stiff grade penalty. The solution for this phase will be released on the next day -- no late assignments will be accepted after solutions are released.

Again, if you have questions regarding the turn-in of this project or problems downloading any of the materials below, please post them on Piazza or ask someone well in advance of the turn-in date. Waiting until the day before it is due to get help is unwise -- you risk not getting a timely response that close to the deadline and will not be given an extension because of such an error.

Additional Information and Supporting Materials:

Instructions for Accessing Remote Repositories
High Level Use Cases for Phase 3

Due Date: March 19, 2017

Weight: 6.0

67-272 :: Phase 4: More Models and Unit Testing

Overview

This phase we will continue our project to develop a foundation for the chess store system. In this phase we will focus our efforts on building out the models for the remaining entities given to you in the documentation provided below and adding all necessary business logic. In addition, you will have to write unit tests for all models. This phase will constitute 11 percent of your final course grade and is broken down into the following three components:

1. Creation of Models: Models for all the entities listed in the documentation below (also handed out in class) need to be created. Below is a specification guide to essential methods each model must have; those methods have to be fully implemented. Any revisions to previous models must be made as specified in the documentation.

2. Unit Testing: Unit tests for all methods in all models must be written and fully passing. We will check to make sure there is 100 percent code coverage for unit tests using the simplecov gem used in class and in lab. All models in this phase need complete coverage. Be aware there are steep penalties for less than 100 percent coverage and no credit at all for less than 90 percent coverage.

3. Coding Style: All code must be appropriately organized. What that means at this stage is the following: Indentation should be used consistently to make code readable. Comments showing organization should be present and explaining difficult code should be used when/if necessary.

List of Projects

Phase 1: Databases and Use Cases

Phase 2: Models and Unit Testing

Phase 3: Basic Views and Controllers

Phase 4: More Models and Unit Testing

Phase 5: Finishing Chess Store

Starting the project

Starter code is available for you on Github. It is identical to the starter code in phase 3 (since we are not doing views and controllers this phase, the same models are sufficient as a starter), but it does have a revised `.gitlab-ci.yml` file that will work for this project.

To get this code, it would be best to clone the existing repository with the following command:

```
git clone https://github.com/profh/ChessStore_Phase_4_Starter.git
```

After this, switch into the project directory and remove the reference to `origin` with the following:

```
git remote rm origin
```

This will stop you from accidentally trying to push changes to Prof. H's repo (which won't be accepted). Now it is recommended that instead you set up a connection to your remote repository on darkknight as soon as you are sent access information on it (remote repos will be created shortly) and begin saving your code there early in your development. There is no limit on the amount of times you can commit code to the remote repository and remember that commiting to your local repository does not automatically mean the remote repository has been updated.

The most important thing is that you **start this phase as early as possible**. There are more models to deal with than phase 2 and many more wrinkles to work through. This will take approximately twice as long to complete as phase 2.

Finishing the project

Your project will be turned in via git -- at 11:59pm on April 9, 2017 we will get a copy of your files in your remote git repository on darkknight.hss.cmu.edu. We will publish instructions on how to access your remote repository, but you should try to access your remote repository early in this phase; if you have not posted any code to the remote repository prior to April 3 and run into access issues in the last few days of the project there will not be much grace given and you can expect a stiff grade penalty. The solution for this phase will be released on the next day -- no late assignments will be accepted after solutions are released.

Again, if you have questions regarding the turn-in of this project or problems downloading any of the materials provided, please ask Prof. H or a Head TA well in advance of the turn-in date. Do not wait until 48 hours or less before it is due to get help as we may not be able to respond in a timely fashion that close to the deadline.

Additional Information. Below are links to additional files references above.

Model Specifications for Phase 4
Database Design for Phase 4
Data Dictionary for Phase 4
ERD for Chess Store

Due Date: April 09, 2017

Weight: 11.0

67-272 :: Phase 5: Finishing Chess Store

Overview

This phase we will finish our project to develop an online system for the chess store. This phase will constitute 11 percent of your final course grade and is broken down into the following components:

1. Implementation of Site Design: An appropriate visual design for the site using CSS and appropriate images should be created. When users log into the application, they should go to a home page that provides all the key information a user would immediately want access to (or have access to it with just one click). An appropriate way for each type of user to navigate throughout the site must be clearly evident and allow easy transitions to other parts of the application. **A lot of the grade this phase will be on the quality of interactions you design for each of the users and how well you implement the design concepts discussed in class.**

As last time, semi-static pages for about, contact, privacy and the like must be in place. No 'lorem ipsum' style text is accepted on any page; text should be appropriate for the page. The student's name must appear on the bottom of every page in the footer (name should be listed as webmaster or designer).

2. Authorization: Administrators as usual have complete authority to do whatever is needed on the site. Other actors will have restricted access as specified in the narrative.

3. Testing: All unit tests (given in the starter code) must be fully working. Since you were given a solution, expect a penalty if this solution is not implemented and any unit tests fail. As for cucumber tests, there are no tests provided this phase in order to maximize your freedom to design. As a result, the weight this phase is considerably less for tests and much more on designing appropriate interactions.

4. Coding Style: All code must be appropriately organized. What that means at this stage is the following: Indentation should be used consistently to make code readable. Comments showing organization should be present and explaining difficult code should be used when/if necessary. All business logic should be in models, not views or controllers.

Starting the project

Starter code is available for you on Github. This starter code has some additional code modules not previously given, so we strongly recommend you start with this as your base and port over styles, views, etc from phase 3 that could be helpful. To get this code, it would be best to clone the existing repository with the following command:

```
git clone https://github.com/profh/ChessStore_P5_Starter.git
```

List of Projects

Phase 1: Databases and Use Cases

Phase 2: Models and Unit Testing

Phase 3: Basic Views and Controllers

Phase 4: More Models and Unit Testing

Phase 5: Finishing Chess Store

After this, switch into the project directory and remove the reference to `origin` with the following:

```
git remote rm origin
```

This will stop you from accidentally trying to push changes to Prof. H's repo (which won't be accepted). Now it is recommended that instead you set up a connection to your remote repository on darkknight soon and begin saving your code there early in your development. There is no limit on a number of times you can commit code to the remote repository and remember that committing to your local repository does not automatically mean the remote repository has been updated.

After cloning the project, run `bundle install` to get any gems needed. After that, populate the database with `rake db:populate` to get a set of realistic records in your database. (We've added `rake db:drop` , `rake db:migrate` , `rake db:test:prepare` to the populate script so anytime you run it, it will completely reset both your dev and test databases.) However, this will only work once you have added authentication support to the user model (and tested it, of course).

Ajax

Ajax can be used wherever it makes sense to improve the user experience. However, there are three places where ajax is required in this project and two are on the `items#show` view. The first use of ajax is related to prices; near the price history table (which is only viewable by managers and admins) there should be an option to add a new price. When this option is selected, a form appears with just a new price box and options for the category; when submitted the form disappears and the current price and price history is updated. The second use of ajax is to adjust inventory level via a new purchase. When a manager or admin makes selects this option, a simple form for a new purchase appears and when completed it will update the inventory level.

The final use of ajax must occur on the shipper's list that appears when the shipper logs in. For each item in an order, when the item is packed and shipped, the shipper clicks a link or button that will mark the item as shipped. Once the database is updated and the shipped_on date is set to the current date, either the order item will disappear or be marked as shipped.

Other notes

A few other notes students should be aware of:

- If a table has no content, then give an appropriate message rather than displaying an empty table.
- No 'geek dates' – format dates in a user friendly fashion and do so for an American (rather than European) audience.
- If the computer should know the date (because it is today, for instance) then users probably shouldn't have to enter the date.
- All tables, lists and select options must be appropriately ordered.
- Appropriate and clear feedback for the user needs to be given to reduce misunderstanding.
- Navigation options appropriate for each type of user should be provided.

Finishing the project

Your project will be turned in via git no later than 4:59pm on May 5, 2017. We will transfer a copy of your files from your remote git repository on darkknight.hss.cmu.edu to the grading system on May 10th at 9:00am. We know that everyone will turn in their applications on time, so we will not bother to check timestamps in this last phase. We will keep access open to the remote repositories and answer questions regarding the project after the due date since those things may prove helpful to students studying for the final exam. Once we start transferring files on May 10th, we will have to cut off student access to the server, so any applications missing or largely defective at that time will get a zero grade.

Again, if you have questions regarding the turn-in of this project or problems downloading any of the materials provided, please ask Prof. H or a Head TA well in advance of the turn-in date. Do not wait until 48 hours or less before it is due to get help as we may not be able to respond in a timely fashion that close to the deadline.

Additional Information. Below are links to additional files references above.

Abilities enumerated for Phase 5

Due Date: May 05, 2017

Weight: 11.0