Computer Science 164: Mobile Software Engineering
Harvard College
Spring 2012

Project 0

HarvardCourses

each milestone's deadline is noon
see `cs164.net/expectations` for each milestone's expectations

Mon	Tue	Wed	Thu	Fri
				2/3 Proposal
2/6 Design Doc, Style Guide				**2/10** Beta
				2/24 Release

Academic Honesty

All work that you do toward fulfillment of this course's expectations must be the work of you and your partner. Collaboration with anyone other than the partner with whom you begin the semester is not permitted unless one of the course's heads approves a change of partner in writing. Partners must contribute equitably to each milestone: you may not implement most or all of some project's milestone and submit it on behalf of your two-person team.

Viewing or copying another individual's work (even if left by a printer, stored in an executable directory, or otherwise exposed) or lifting material from a book, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student or soliciting the work of another individual. Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this course that you have submitted or will submit to another. Nor may you provide or make available solutions to projects to individuals who take or may take this course in the future. Moreover, submission of any work that you intend to use outside of the course (*e.g.*, for a job) must be approved by the staff.

You may read and comment upon classmates' code toward fulfillment of projects' code reviews but only for classmates whose code is assigned to you by the course's staff for review. You may integrate ideas and techniques that you glean from your reviews of classmates' code and from classmates' reviews of your code into your own work, so long as you attribute those ideas and techniques back to your classmates, as with comments in your own code. As for classmates beyond your own partner and those with whom you're involved in reviews, you may discuss projects, including designs, but you may not share code. In other words, you may communicate with those classmates in English, but you may not communicate in PHP, JavaScript, or Objective-C. If in doubt as to the appropriateness of some discussion, contact the course's heads.

You may turn to the Web for instruction beyond the course's lectures and labs, for references, and for solutions to technical difficulties, but not for outright solutions to projects or portions thereof. However, failure to cite (as with comments) the origin of any code or technique that you do discover outside of the course's lectures and labs (even while respecting these constraints) and then integrate into your own work may be considered academic dishonesty.

All forms of academic dishonesty are dealt with harshly. If the course refers some matter to the Administrative Board and the outcome for some student is *Admonish*, *Probation*, *Requirement to Withdraw*, or *Recommendation to Dismiss*, the course reserves the right to impose local sanctions on top of that outcome for that student that may include, but not be limited to, a failing grade for work submitted or for the course itself.

Help.

Help is available throughout the week at `http://help.cs164.net/`, and we'll do our best to respond within 24 hours. But do turn first to your partner with bugs!

Getting Started.

□ First dive into HTML5 at `http://diveintohtml5.info/`, a free online book written in, well, HTML5! You won't need to leverage all of HTML5's features for this or future projects, so feel free to skim or skip sections that aren't of interest.

□ Next curl up with the free O'Reilly book at `http://ofps.oreilly.com/titles/9780596805784/`. At a minimum, peruse chapters 1 through 3 plus 5.

□ Now spend some time with `https://developer.mozilla.org/en/JavaScript/Guide`, particularly sections 8 through 13 under **Quick Table of Contents**.

□ If unfamiliar with XML and/or parsing XML with PHP, read up on both at `http://www.ibm.com/developerworks/xml/library/x-xmlphp1/` and `http://www.ibm.com/developerworks/xml/library/x-xmlphp2/`.

□ If unfamiliar with git (and specifically merges), learn how to version files like ~~a boss~~ Tommy via the seminar at `https://manual.cs50.net/Seminars#Git_Magic:_Versioning_Files_Like_a_Boss`. You might also find helpful the Git Community Book at `http://book.git-scm.com/`.

□ So that you have a place to store proposals, style guides, design docs, and code this term, head to `https://bitbucket.org/plans` and sign up for a **FREE** account. Be sure to use a `.edu` address so that you're automatically upgraded to an unlimited student plan.[1] Even if you already have an account on some other repo-hosting site (*e.g.*, GitHub), do sign up for Bitbucket so that repos can be shared easily among classmates and staff.

Once signed up, inform your partner of your Bitbucket username. Beware typos, lest your partner share his or her code with some random person on the Internet!

[1] If you already have an existing account, you're welcome to use that for the course. You can apply to have it upgraded to an unlimited student plan at `http://www.atlassian.com/software/views/bitbucket-academic-license.jsp`.

□ Now it's time to download some software.

- □ If you don't have it already, install the latest version of Google Chrome from `http://www.google.com/chrome`, then install the Window Resizer extension from `https://chrome.google.com/webstore/detail/kkelicaakdanhinjdeammmilcgefonfh`.
- □ If you don't have it already (as you might if in CS51), install version 3 of the CS50 Appliance, per the instructions at `https://manual.cs50.net/Appliance#How_to_Install_Appliance`.[2,3] **Be sure to use VMware Fusion or VMware Player this term, not VirtualBox**.[4] If you've not used the CS50 Appliance before, see `https://manual.cs50.net/Appliance#How_to_Use_Appliance` to learn how to use it!
- □ If you have a Mac running Lion, install Xcode 4.2.1 from `http://itunes.apple.com/us/app/xcode/id448457090?mt=12`. Technically, you don't need a Mac for the course until Mon 3/19, but Xcode comes with iOS Simulator, which might prove handy for testing in the short term. If you do have a Mac, know that you can use Snow Leopard for the course, but Apple only makes Xcode available to Snow Leopard users through its $99 developer program,[5] whereas Lion users can download Xcode for free from the Mac App Store. Granted, Lion itself costs $29.99.

□ Now that you have version 3 of the CS50 Appliance installed (along with VMware Fusion or VMware Player), start the appliance. Select **Menu > Internet > Google Chrome** inside the appliance and then visit your favorite website to confirm that the appliance has Internet access. (Suffice it to say your own computer must too!) Then open a terminal inside the appliance, per `https://manual.cs50.net/CS50_Appliance_3#How_to_Open_a_Terminal`, or SSH from your own computer to the appliance, per SSH from your own computer to the appliance, per `https://manual.cs50.net/Appliance#How_to_SSH_to_Appliance`, and update the appliance by executing the command below.[6]

```
sudo yum -y update
```

Now configure the appliance for CS164 by executing the command below.

```
sudo yum -y install cs164
```

If either command fails, try it again after restarting the appliance (as via **Menu > Log Out > Restart**). If still no luck, turn to your partner or `help.cs164.net` for a hand!

[2] If you took CS50 in Fall 2011, you have version 2.3, which is too old for CS164 and cannot be updated to version 3 via `yum`. You'll need to download version 3.

[3] If you do already have version 3 of the CS50 Appliance for CS51, you can use that same installation for CS164; you don't need to download a second copy.

[4] Though free, VirtualBox proved a bit unstable for some students in Fall 2011.

[5] `http://developer.apple.com/programs/ios/`

[6] In version 3 of the appliance, `sudo` no longer requires John Harvard's password.

□ If unfamiliar with virtual hosts (vhosts), acquaint yourself at http://en.wikipedia.org/wiki/Virtual_hosting.

Then, in a terminal window inside the appliance, execute

```
sudo gedit /etc/hosts
```

and add the following line at the bottom of the file that opens, then save and quit gedit.[7]

```
127.0.0.1 project0
```

Recall that sudo executes a command as the super-user (*i.e.*, root), which is necessary in this case, since /etc/hosts is only writeable by root. The line you just added ensures that project0 will "resolve" (as via DNS) to 127.0.0.1, which is the appliance's "loopback address."

Next, create a directory called vhosts in your (well, John Harvard's) home directory. Then create a project0 directory within vhosts. Then create an html directory within project0. Then chmod all three, plus your home directory, 711.

Now create a file called index.php inside of ~/vhosts/project0/html/ containing the below:

```
<?= 'hi' ?>
```

Then chmod the file 600 (though it should be already) and visit http://project0/ with Chrome inside the appliance. Be sure to type the http://, else you'll end up Googling "project0". You should be greeted with hi. If you instead see some error, best to retry these steps![8]

If curious as to why all this works, take a peek at /etc/httpd/conf.d/cs164.conf. That file maps all of the appliance's virtual hosts to /home/jharvard/vhosts/ so that creating a new virtual host called, say, foo inside the appliance is as easy as creating /home/jharvard/vhosts/foo/ and /home/jharvard/vhosts/foo/html/ and editing /etc/hosts so that foo resolves to the 127.0.0.1.[9] Thanks to virtual hosts, each of your projects can appear to live in the root of a webserver (as opposed to some subdirectory), just like a real site!

[7] No need to append project0.localdomain to that line.

[8] If you instead see bye, you really did something wrong.

[9] You can also access the appliance's virtual hosts from a browser on your own computer if you'd like, but you'll first need to edit /etc/hosts (if running Mac OS or Linux) or C:\Windows\system32\drivers\etc\hosts (if running Windows) on your own computer similarly. Just be sure to replace 127.0.0.1 with the appliance's IP address, which is displayed at all times in the appliance's bottom-right corner. To edit these files with a text editor, you'll likely need to invoke sudo (if running Mac OS or Linux) or **Run as Adminstrator** (if running Windows).

□ **Okay, for the sake of discussion, we need to start calling you or your partner Alice and the other of you Bob. Decide who will be who, then carry on!**

□ **Alright, these next steps only Alice should perform**.

Hi, Alice. Visit `http://project0/phpMyAdmin/` with Chrome inside the appliance. Authenticate as **jharvard**, whose password is **crimson**.

Click **Databases** and, under **Create new database**, input **jharvard_project0**, then click **Create**.[10]

□ **And only Alice should perform these steps too**.

Log into your Bitbucket account and create a new, private repo as follows:

□ Ensure that **Create new repository** is highlighted (in dark blue).
□ Input a value of **project0** under **Name**.
□ Ensure that **Private** is checked.
□ Ensure that **Git** is selected under **Repository type**.
□ Check both **Issue tracking** and **Wiki** under **Project management**.
□ Select **PHP** under **Language**.
□ Input a value for **Description** and/or **Website** if you'd like.
□ Click **Create repository**.

You should then find yourself at a page whose URL is `https://bitbucket.org/alice/project0`, where `alice` is your actual Bitbucket username. **Ignore the instructions about `clone`**. Click the **Admin** tab at top, then click **Access management** at left. In the text field under **Users (1)**, input your partner's Bitbucket username, then click **Admin** at right. Finally, input **cs164** into that same text field, then click **Admin** at right. Your partner and CS164's staff should now have access to your repository. Your partner can confirm as much by visiting `https://bitbucket.org/alice/project0`, where `alice` is your actual Bitbucket username.

□ **Both Alice and Bob should now perform these steps, with Alice using her Bitbucket account and generating her own SSH keypair, and with Bob using his Bitbucket account and generating his own SSH keypair. Alice and Bob need not (and should not) share each other's private key.**

Open a terminal inside the appliance or SSH to it from your computer. Then execute the following command to generate an SSH key pair (one public key, one private key):

```
ssh-keygen
```

Hit Enter at each of the prompts that appears (unless you'd prefer to secure the private key with a keyphrase). If you then execute

```
ls ~/.ssh/
```

[10] Note that the name of any database that you (well, John Harvard) creates must begin with **jharvard_**.

you should see that you now have (among others) files called id_rsa (your private key) and id_rsa.pub (your public key). Execute

```
gedit ~/.ssh/id_rsa.pub
```

then highlight and copy the contents of that file (which happens to be your public key). Visit https://bitbucket.org/account/ and paste the key into the text field below **SSH keys**, then click **Add key**.

☐ Okay, **only Alice should perform these steps**.

Download version 2.1.0 of CodeIgniter from http://codeigniter.com/download.php inside the appliance, as with Chrome or by executing a command like the below in a terminal window.

```
wget http://codeigniter.com/download.php
```

Unzip the file you just downloaded, as by executing the command below in John Harvard's **Downloads** folder.

```
unzip CodeIgniter_2.1.0.zip
```

Within the folder you just unzipped (CodeIgniter_2.1.0) should be a few files and directories: move application and system to ~/vhosts/project0/, then move index.php to ~/vhosts/project0/html/.
Open up index.php with a text editor and change the value of $system_path from 'system' to '../system' and the value of $application_folder from 'application' to '../application'. Then save your changes.

Revisit http://project0/ with Chrome inside the appliance (reloading if necessary), and you should see **Welcome to CodeIgniter!**. If not, best to retrace your steps!

Once you do, execute the following commands to add your code (well, CodeIgniter's) to a local repository, where Alice is your own first and last name (in real life) and alice@example.com is your own email address (in real life):

```
cd ~/vhosts/project0/
git config --global user.name "Alice"
git config --global user.email "alice@example.com"
git init
git add --all
git commit -m "Initial commit"
```

Then execute the below to add a "remote" for your Bitbucket repo, where `alice` is your own Bitbucket username:

```
git remote add origin git@bitbucket.org:alice/project0.git
```

Now push your code to that remote:

```
git push -u origin master
```

□ Lastly, **only Bob should perform these steps**.

First delete the `project0` directory you created earlier whilst testing (**assuming you haven't done anything important in there**), as by executing the command below in a terminal window, inputting **y** as prompted:

```
rm -r ~/vhosts/project0/
```

Next execute the below, where `Bob` is your own first and last name (in real life) and `bob@example.com` is your own email address (in real life):

```
git config --global user.name "Bob"
git config --global user.email "bob@example.com"
```

Next execute the below, where `alice` is Alice's own Bitbucket username, not yours:[11]

```
cd ~/vhosts/
git clone git@bitbucket.org:alice/project0.git
```

If you now execute

```
ls
```

you should see that you have copies of the files and directories that Alice pushed to your shared Bitbucket repo. Proceed to `chmod` things just as Alice did earlier. Then confirm that your own vhost works by visiting `http://project0/` with Chrome inside the appliance.

□ Woo hoo! You're now on the same page.

□ Henceforth, anytime you make changes to code that you'd like track in version control, execute the below (or similar, if more experienced with `git`) inside of `~/vhosts/project0/`:[12]

```
git add --all
git commit -m "a description of the changes you've made"
git push
```

[11] Recall that your **project0** repo technically lives in Alice's Bitbucket account, but you nonetheless have admin privileges for it.

[12] If you omit the `-m` flag when committing, `git` will prompt you for a commit message with `nano`. Once you've provided said message, you can save and quit `nano` with **ctl-x**, **y**, then **Enter**.

Your partner can then download those updates with the below, whilst in his or her own `~/vhosts/project0/`:

```
git pull
```

Beware, though: if you've both made changes to the same line of code in the same file, `git` may prompt you to resolve the conflict.

Alice, note that your MySQL database will not be pushed to Bitbucket by default. Odds, are, though, Bob will want a copy of it. And both of you will want to pull any changes the other makes to its tables. See `cs164.net/hooks` to learn how to share MySQL databases via git "hooks."

□ To learn more about CodeIgniter, peruse `http://codeigniter.com/user_guide/`, particular its **Introduction** and **Tutorial**. (Click **TABLE OF CONTENTS** up top to find both.) To learn more about jQuery Mobile, visit `http://jquerymobile.com/demos`/ and examine those pages' source code. And if unfamiliar with jQuery, review `http://docs.jquery.com/How_jQuery_Works`!

Specification.

□ Here we go!

□ Your challenge for this project is to implement a mobile web app with which users can shop for FAS courses. The overall design and aesthetics of this app are ultimately up to you, but we require that your app meet some requirements. **All other details are left to your own creativity and interpretation.**

Note that, after submission of your beta, your TF will decree (not unlike a pointy-haired boss) that you and your partner must improve some feature's design or implement some new feature altogether in time for your app's release. Try, then, to anticipate future requests, features that might belong in an app like this one. The better your design now, the easier it will be to support future features! We'll still try, though, to trip you both up!

Feature Requirements.

□ Your app's UI should be designed for a smartphone whose width is defined by `device-width`; its actual resolution might be anywhere from 320×480 to 760×1280.
□ Your app must support, at least, Spring 2012 courses.
□ Your app must enable users to browse courses by department; interpret "department" as you see fit.
□ Your app must enable users to browse courses by Gen Ed area.[13]
□ Your app must enable users to search for courses by keyword, whereby those keywords may appear in courses' catalog numbers, titles, descriptions, and/or instructors' names.
□ Your app must enable users to search for courses by day and time.

[13] RIP Core.

☐ Your app must enable users to see courses' catalog numbers, titles, departments, descriptions, instructors, locations, and days and times.

☐ Your app must enable users to add courses to lists called **Courses I'm Shopping** and **Courses I'm Taking** (or similar); you needn't support custom lists. Your app must enable users to browse each of those lists.

☐ Your app must enable users to browses courses they've recently viewed, as via a special list called **Recently Viewed**; we leave it to you to define "recently."

Technical Requirements.

☐ Your app must be implemented with HTML5, JavaScript, and PHP 5.3.

☐ Your app must be developed and/or tested within `/home/jharvard/vhosts/project0/` in version 3 of the CS50 Appliance. You're welcome to develop it in some other environment, so long as it ultimately works if installed in that directory (and `chmod`'d appropriately).

☐ If you'd like to test your app via the Internet with an actual mobile device, you're welcome to configure a vhost on `cloud.cs50.net`, per `https://manual.cs50.net/vhost`. Visit `https://cloud.cs50.net/` to request a CS50 Cloud account as needed.

☐ You may use any IDE or text editor you'd like to develop your app, but `nano` and `gedit` are discouraged. Among text editors for Linux, we recommend `vim` and `emacs`. Among text editors for Mac OS, we recommend TextWrangler. Among text editors for Windows, we recommend Notepad++. Among IDEs for all OSes, we recommend NetBeans.

☐ Your app must use jQuery Mobile (and jQuery) for its client-side framework.

☐ Your app must use CodeIgniter for its server-side framework.

☐ Your app may use third-party libraries and plugins (for CSS, JavaScript, and/or PHP) in addition to, but not instead of, jQuery Mobile and CodeIgniter, so long as their sources are cited, as with comments.

☐ Your app must adhere to an MVC architecture (as it should by nature of CodeIgniter).

☐ Your HTML5 must be well-formed but it need not be considered valid by the W3C's validator (which dislikes browser-specific tags).

☐ You must use `http://cdn.cs164.net/2012/spring/projects/0/courses.xml` for your catalog of courses. (We leave it to you to figure out its structure.) It's 10MB, which won't be fun for mobile users on slow connections to download, so odds are you'll want to chop it up into multiple files, convert it to JSON, and/or load it into SQLite or MySQL tables (with indexes). You must not use CS50's HarvardCourses API.

☐ Your app must store users' lists server-side in a MySQL database or client-side in a SQLite database or in HTML5 `localStorage`.

☐ You need not require users to log in.

☐ You must use `git` and Bitbucket for version control.

☐ Under no circumstances should we be able to trigger runtime errors in your PHP or JavaScript code. Be sure that you handle unwanted inputs and HTTP failures elegantly, as by reporting such errors or silently handling. Under no circumstances should your code trigger errors in a browser's viewport or console.

Decisions.

□ Ultimately, we leave it to you and your partner to decide how to design (and implement!) this project's requirements. But allow us to facilitate conversation with some rhetorical questions.

- □ Who will do what?
- □ When will you do it? Assume that everything will take longer than you expect!
- □ How will you integrate `courses.xml` into your app? If you convert it to JSON, how will you search it? If you import it into SQLite or MySQL tables, what will your tables' primary and/or foreign keys be? What fields should you index for speed?
- □ Will searches be executed client-side or server-side?
- □ How will you make the app user-friendly? Odds are users won't want to scroll through hundreds of courses. Should you support autocomplete?
- □ How will users search for courses by day and time? What UI for such would real students find useful?
- □ How will users add courses to lists?
- □ Should you use MySQL, SQLite, or `localStorage` for users' lists?
- □ How will you design your app with future (unknown) features in mind?
- □ What else?