IERG4210 Web Programming and Security (2015 Spring)
Assignment Marking Guidelines

Revision History

v3.0	Feb 5	Released Phase 3 Requirements
v2.0	Jan 28	Released Phase 2 Requirements
v1.0	Jan 16	Created this document; Released Phase 1 Requirements

General Guidelines

The assignment is designed to let students practice what they have learned in the course. Students must be aware of web application security throughout the web development. The whole assignment is split into 7 phases, leading all the way to a fast, secure, and user-friendly shopping website upon completion. Students can take a real-world website, walmart.com, as a reference. In the assignment, students are expected to understand and apply proper security design principles and programming skills, regardless of what libraries (e.g., jQuery) the students would like to use. The marking checklist in the next page is written in a minimal-viable and result-oriented basis, and thus students can unleash their creativity in building more features. For detailed guidance, students should refer to both tutorial and lecture notes.

Submission Policy

Each student is required to maintain their source code and any resources (e.g., images, css and js files) in an assigned private repository **shop[01-80]** at github.com/ierg4210. While latest changes are always maintained in the *master* branch, students are required to branch out a snapshot titled as **phase[1-7]** from its *master* for each phase. Hence, TAs will pull from the particular branch and only take that into account for inspection. Technical details can be found in Tutorial 2.

Each phase is associated with a firm submission deadline.

- *Early Submission Incentive* – For every 48-hour advanced submission in one phase, the deadline for phase 5 or 6 can be extended by 24-hour, and no part thereof is accepted. For instance, submitting 100 hours before the phase 1 deadline will gain an extension of 48 hours for either phase 5 or 6 deadline of your choice.
- *Late Submission Penalty* – Late submission will incur deduction of $(10\%)^{1/n}$ from your scored points, where n is the round-up number of days delayed (e.g. 9 hrs late → 10%, 25 hrs late → ~31.6%, 49 hrs late → ~46.5%, and so forth).
- *Interim Demonstration* – Students' submissions will be randomly sampled by TAs for inspection. If a student is found unable to complete 70% of the requirements in any single phase from 1 to 5, s/he is required to see TA, and will be arranged to give an interim demonstration (time and venue TBD). If the student fails to complete 70% of the requirements by the time his/her interim demonstration is given, the case will be escalated to the department for resolution. Students capable of meeting deadlines and 70% of the requirements can safely ignore this policy.
- *Final Demonstration* – Students will sign up for a timeslot to demonstrate their websites to a marker, who will then grade it according to the checklist. The marker will further evaluate the student's understanding with two questions.

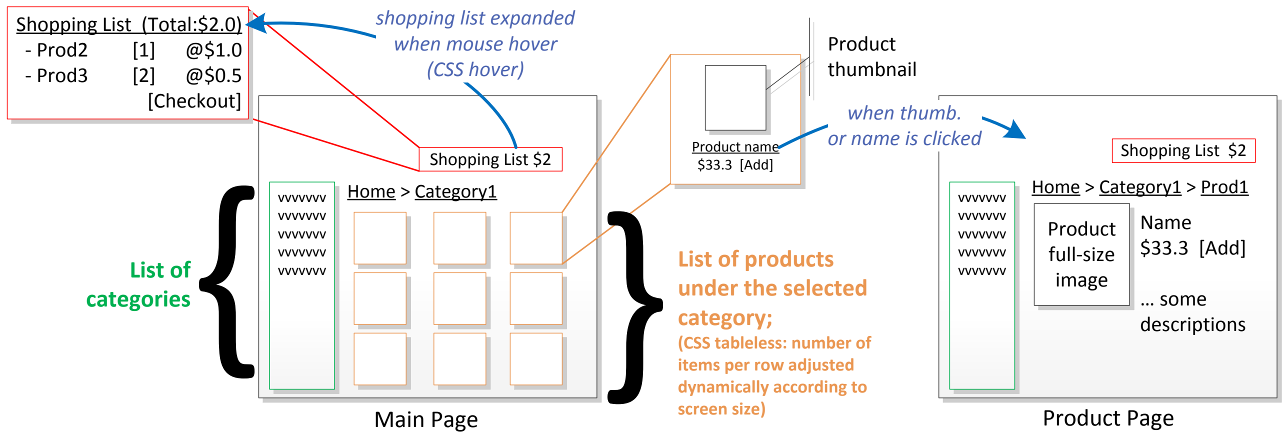
Honesty in Academic Work

CUHK places very high importance on honesty in academic work submitted by students, and adopts a policy of *zero tolerance* on cheating in examinations and plagiarism. Students are NOT allowed to submit anything that are plagiarised. Therefore, we treat every assignment our students submit as original except for source material explicitly acknowledged. We trust that students acknowledge and are aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/.

IERG4210 WEB PROGRAMMING AND SECURITY (2015 SPRING)
ASSIGNMENT MARKING CHECKLIST V3.0

PHASE 1: LOOK AND FEEL (DEADLINE: JAN 26, 2015, 5PM) (SUBTOTAL: 12%)

A designer has drafted a layout as follows, which outlines some fundamental features of a shopping website. In this phase, you will create a mock-up by hardcoding the website with **dummy categories and products**.



1. HTML: Make use of semantic HTML <u>throughout the whole assign.</u> _____ / 1%
   - <header>, <nav>, <footer>, <article>, <section>, <ul>, <li>
2. CSS: Clean separation of HTML, CSS and JS code and files <u>throughout the whole assign.</u> _____ / 2%
   - No inline CSS and JS are allowed
   - No styling in HTML, e.g. <center>, <div align="center">, etc
   - Tolerance: < 5 exceptions
3. *Main page* demonstrates the use of "CSS tableless" *product list* _____ / 2%
   - Each product has <u>at least</u> its own thumbnail, name, price and *addToCart* button
   - When the thumbnail or name is clicked, redirect to the corresponding product page
4. *Main page* demonstrates the use of "CSS hover" *shopping list* _____ / 2%
   - When displayed, it will cover any elements behind
   - Input boxes are used for inputting quantity of each selected product
   - A checkout button is used to submit the list to Paypal
   - The shopping list is displayed in both main and product pages
5. *Product page* provides product details _____ / 2%
   - To show a full-size or bigger image, name, description, price, and *addToCart* button
6. Both main and product pages should include a navigation menu _____ / 2%
   - e.g., Home or <u>Home</u> > Category1 or <u>Home</u> > <u>Category1</u> > Product1
   - Those underscored are hyperlinks that redirect users to an upper hierarchy
7. Branch out **phase1** in your repository, where TAs can checkout for inspection _____ / 1%

PHASE 2: TESTING ENVIRONMENT SETUP (DEADLINE: FEB 4, 2015, 5PM) (SUBTOTAL: 11%)

In this phase, you are required to setup a local development environment and also a remote deployment environment. Some guidance will be given in tutorial 4.

1. Create a web application and server using Node.js
   - Initialize a new node project, and install the following npm packages _____ / 1%
     - as dependency: `express` and `express-handlebars`
     - as development dependency: `supervisor`
   - Hosting your server locally _____ / 1%
     - Use `supervisor` to run your app locally to avoid manual restarting upon code changes

        - Server accessible at http://localhost[:3000] or any port number
    - Serve static files using the express.static middleware _____ / 2%
        - Static image files accessible through http://localhost[:3000]**/images/**
        - Static JavaScript files accessible through http://localhost[:3000]**/lib/**
        - Static CSS files accessible through http://localhost[:3000]**/css/**
    - Serve dynamic pages with a *server-side* JS templating engine _____ / 3%
        - Based on the simple basic example of ExpressHandlebars (sample code)
          - Move the common UI code from your two HTML into a main template layout
          - Hence, core contents of your pages will be combined with the main template
        - The *main page* accessible through http://localhost[:3000]/
        - The *product page* accessible through http://localhost[:3000]/**product**
2. Deploy your code to Amazon Elastic Beanstalk** (EB) _____ / 4%
    - Create a EB environment for Node.js (application name: shopXX-ierg4210)
    - Deploy your application to the EB environment#
    - Remotely accessible through the given application URL
    - Branch out **phase2** in your repository, where TAs can checkout for inspection
      - `.gitignore` configured to exclude committing any installed node and virtualenv packages

** *This will incur charges:*
(1) Apply the free US$100 AWS credit code, to be distributed by TA
(2) After the first deployment, configure to use single instance to save cost during development
(3) Terminate any unused resources (during/after this course). You're responsible for charges beyond your free quota

Phase 3A: Backend Portal and Management (Deadline: Feb 18, 2015, 5pm) (Subtotal: 8%)

In this phase, you will build a backend portal for administrators to manage your products.

1. Create a database with the following structures _____ / 1%
    - A table for *categories*
        - Required columns: *catid (primary key, unique), name*
        - Data: at least 2 categories of your choice
    - A table for *products*
        - Required columns: *pid (primary key, unique), catid, name, price, description*
        - Data: at least 2 products for each category
2. Create a backend management portal for admin
    - Design several HTML forms to add, change*, and delete a category in DB _____ / 2%
        - Input field for category's name
    - Design several HTML forms to add, change*, and delete a product in DB _____ / 5%
        - Dropdown menu to choose and change the category (using <select>)
        - Input field for product's name
        - Input field for product's price (supporting 2 decimal places)
        - Textarea for product's description
        - File field for product's photo
          (Format: jpg/gif/png, Size: <=10MB, Name: *pid*.[jpg|gif|png])

  * The original values must be restored back to the inputs before a change (display the photo nearby)

Phase 3B: Frontend Presentation (Deadline: Feb 27, 2015, 5pm) (Subtotal: 16%)

In this phase, you will implement the shopping cart that allows users to shop around your products. This phase is designed to let you practise data presentation and AJAX programming.

1. Populate the *main page*'s contents from DB with Handlebars or ExpressHandlebars
    - Populate the *category list* from DB _____ / 1%
    - Based on the category picked by user, populate the corresponding *product list* from DB _____ / 3%
        - Reflect catid in the URL (i.e., /?catid=[x], or simply /[x])
        - The corresponding product list is shown upon accessing the new URL in a new tab
2. Populate the *product page*'s contents from DB with Handlebars or ExpressHandlebars _____ / 2%

- Display the details of a product according to its DB record

3. Using JavaScript, dynamically update# the *shopping list*
   - When the *addToCart* button of a product is clicked, add it to the shopping list _____ / 1%
     - Adding the same product twice adds up quantity, do not display two rows of record
   - Once a product is added,
     - Users are allowed to update its *quantity* and delete it with a number input, or two buttons for increment and decrement _____ / 1%
     - Store its *pid* and *quantity* in the browser's `localStorage` _____ / 2%
     - Get the *name* and *price* over AJAX (with *pid* as input) _____ / 3%
     - Calculate and display the total amount at the client-side _____ / 1%
   - Once the page is reloaded, the *shopping list* is restored _____ / 2%
     - Page reloads when users browse another category or visit the product detail page
     - Populate and retrieve the stored products from the `localStorage`

# The whole process of *shopping list* management must be done without a page load

PHASE 4: SECURING THE WEBSITE (DEADLINE: TBD) (SUBTOTAL: TBD%)

In this phase, you will protect your website against many popular web application security threats.

1. No XSS Injection and Parameter Tampering Vulnerabilities in the whole webapp
   - [UX Enhancement Only] Proper and vigorous client-side input restrictions for all forms _____ / 1%
   - Proper and vigorous server-side input validations sanitizations and for all forms _____ / 2%
   - Proper and vigorous context-dependent output sanitizations using TBD _____ / 2%
2. More requirements will be released

PHASE 5: SECURE CHECKOUT FLOW (DEADLINE: TBD) (SUBTOTAL: TBD%)

This is a toughest phase, yet the most critical one to escalate your website to a professional level.

1. More requirements will be released

PHASE 6: EXTENSIONS (DEADLINE: TBD) (SUBTOTAL: >5%. TBD, BONUS: __% MAX)

In this phase, you can choose any combinations of the following items to implement. At most _% bonus will be awarded.

1. Support multi-level categories for products (should update both frontend and backend) ______ / 3%
2. Support pagination / AJAX infinite scroll when browsing products in the main page ______ / 3%
3. Using AJAX when browsing categories and products in the main page ______ / 4%
   - Use location.assign to reflect the state of current view
   - The corresponding view is shown upon accessing the new URL in a new tab
4. Apply Search Engine Optimized (or user-friendly) URLs on frontend ______ / 2%
   - Include the name of categories and products into the URLs:
     e.g. /[catid]-[catname]/ for browsing products under the category [catid]
     e.g. /[catid]-[catname]/[pid]-[pname]/ for browsing product details of [pid]
2. Support HTML5 Drag-and-drop image upload in the admin panel ______ / 2%
   - Create and Highlight a dropping zone when a file is being dragged to the zone
   - Display a thumbnail (i.e. smaller width and height) if the dropped file is an image; reject it otherwise
3. Support AJAX image upload in the backend portal ______ / 3%
4. Support automatic image resizing for product images ______ / 3%
   - When a large image is uploaded, the image is automatically resized to produce a thumbnail image
   - In the main page, display thumbnails. In the product description page, display the large image.
5. Subject to changes upon students' requests and upon releasing requirements for more phases

PHASE 7A: SELF HACKING (DEADLINE: TBD) (SUBTOTAL: __%)

1. More requirements will be released

PHASE 7B: PEER HACKING (PERIOD: TBD) (SUBTOTAL: __%, BONUS: __% MAX)

1. More requirements will be released

FINAL Q&A (SUBTOTAL: -75%)

- Random Question 1: Code-related __________%
- Random Question 2: Conceptual or Code-related __________%

SID: ________________________________ TOTAL: ______ / 100%

MARKER RESPONSIBLE: ____________