

Project Milestone 2 (20%)

Due at 17:00 PM on Friday, 1st May 2020 (Week 8)

Project Milestone 3 (20%)

Due at 17:00 PM on Friday, 29th May 2020 (Week 12)

Marking criteria

- All work on this assignment is to be your own **individual** work. Using the code supplied by course staff or released on the Internet is acceptable. However, you should understand the underlying mechanism of each implemented feature in your project. **You may lose marks if you cannot explain the code to the tutors clearly.**
- You must deploy your project to a **remote server** (e.g., UQ zone, AWS) to get it marked.
- Your project must be implemented in a 3-tier architecture, which means communications among front-end, back-end, and database are expected.
- You must complete **all compulsory features** (refer to Appendix 1) in your project and demonstrate them in either Milestone 2 or Milestone 3 (i.e., final submission). Otherwise, the final marks for your entire project will be capped at **20%**.
- Besides the compulsory features, you are encouraged to creatively design and develop innovative features which are appropriate for your project purpose. Refer to Appendix 2 for sample features with different credits (marks) according to their difficulty levels.
- The final marks you will obtain in Milestone 2 will be the total marks for the elective features that are successfully implemented in your project. Please be aware that the final marks will be capped at 20%. Thus, you can select only a subset of completed features for marking and leave the remainder to Milestone 3.
- Note that, one features can be only marked once. E.g. the registration for admin and users is consider one feature, due to the similarity of the implementation.

Blackboard Submission

- You should compress all your source code files into a zip file and submit it to the blackboard at Assessment - Milestone 2 Submission before the deadline. A penalty will be applied to the late submission.
- The file should be named using the following format: “xxxxxxx_milestone2.zip”, (replace xxxxxxx with your Student ID, e.g., s1234567_milestone2.zip)
- Only your **submitted version** will be marked.

Appendix 1 - Compulsory Features

1.1. Login

- Server-side validation is required (e.g., informing the user if the username is not registered in the database or if the password is incorrect).
- Each page should reflect the user's login status (e.g., profile/logout button should appear only when the user has logged in).
- Some content should only be visible for logged-in users.
- Choose an appropriate HTTP request method (e.g., GET, POST, PUT, etc.) for client-server communication.

1.2. Registration

- Username and email address should be checked to be unique.
- Password strength should be checked.
- The registered user data should be saved in the database.
- Choose an appropriate HTTP request method (e.g., GET, POST, PUT, etc.) for client-server communication.

1.3. User Profile

- Create a user profile interface which is only accessible when the user has logged in.
- All the user's information should be fetched from the database.

1.4. Cookies

- Use cookie in at least one of features you implemented

1.5. AJAX

- Use AJAX in at least one of features you implemented.


Appendix 2 - Elective Features (applied to both Milestone 2 & 3)

1. Basic Level (2% each)

- Remember me: retain user's login details even after session expired.
 - You must use Cookie
- Maintain scroll position of large HTML page when client returns
- Favorites, voting or rating
- Search box autocompletion
 - Refer to the google search box
- Adding items (e.g., dishes, events, goods, pictures)
- Bidding items
- Writing comments/reviews
 - Allow unregistered users to comment using their IP addresses as ID (+1%)
- User profile updating (e.g., email, phone number)
- Display the user's current location on a map
 - You must use the user current location information for at least one kind of service in your system, e.g., recommending nearby stores based on the current location
- Image processing (e.g., resize, compress and add watermarks to images)
- Third party API integration (e.g., sending an SMS to verify phone number using SMS API)
 - Google Map, Email, Calendar are excluded
 - Item/video/event recommendation (e.g., google recommendations api)
- Basic file uploading: (e.g., profile image, food pictures, thumbnails)
- Using drag and drop to choose files for uploading (prerequisite: basic file uploading)
- Multiple files uploading at the same time (prerequisite: basic file uploading)
- Web Security (e.g., captcha, password encryption, data sanitizing)
 - 2% for each
- Online Payment Integration (e.g., PayPal, Stripe)
- Image and PDF Manipulation (e.g., generating a receipt)
- Social media integration (e.g. post to Facebook)
- Automatic logout when user inactivity
- Anonymous post/bid, allow user to post/bid anonymously
 - Being able to identify different anonymous users in your system
 - Refer to the anonymous posts on Piazza
- Infinite scrolling (or load more)
- Using reCAPTCHA to enhance web security
- Secure data transaction (end to end encryption, user-to-user) (e.g., Zend framework)
- Create/schedule calendar events
 - Google Calendar API
 - Synchronize Google Calendar data into your system

2. Intermediate Level (3% each)

- Item Searching (e.g., images, meals or items, depending on the topic)
 - Each resulting search item should contain a link which redirects to a detail page
 - fuzzy search, e.g., type "giigle" can be recognized as "Google" by the system (+1%)

- Search filters, e.g., searching items in one specific category group, searching items in a specific pricing range. (+1%)
- Email Verification
 - After registration, an email should be sent to the user for the email verification
 - This feature could be implemented using a random token or a verification code
 - The user should be able to see the verification status, i.e. email verified or not
-  Forgot Password
 - Take account of security concerns.
 - This feature could be implemented using reset tokens or secret questions
- Shopping Cart/Wishlist/Calendar
 - The information should be stored in the database
 - Users should be able to add and remove items from the cart/wishlist/calendar.
 - You can only choose to implement one and collect marks
- Adopt OAuth to allow unregistered users to login
 - Allow users to use their social network account to login with registration
 - Store user information into the database
- Self-implemented Secure data transaction (end-to-end encryption, user-to-user)
 - Develop your own encryption method
 - Secure all communication between user with random generate key pairs

3. Advanced Level (4% each)

- Data Mining (e.g., association rule mining)
- Management Dashboard (statistical information)
 - demonstrate at least four kinds of statistical analysis graphs, e.g., bar chart, pie chart, line chart, sunburst chart
- Web Push Notification (e.g., when the item is sold, when the meal is delivered)
 - Refer to https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API/Using_the_Notifications_API
- Recommendation Algorithm (statistic based)
- Computer Vision Algorithm (e.g., detecting objects in images)
- Real-time Chatbox (self-implemented)
- Setup proxy server
- Chatbots (e.g., alexa)
- **Others at the same difficulty level**

Note that, if you are unsure whether your elective features can be considered as basic or advanced ones, feel free to contact tutors for confirmation.

4. User Interface Design

To obtain marks for UI design, your project must meet all the marking criteria (visual design, HTML usage, CSS usage and navigation) in a column below. E.g., you will get 1% if your website UI meets all the criteria listed in the basic-level column.

	Basic Level (1%)	Medium Level (2%)	Advanced Level (3%)
Visual Design	<ul style="list-style-type: none">• Basic style guide provided with basic rules for application to site elements.• A basic illustration of the visual design of the site.	<ul style="list-style-type: none">• Style guide provided with basic rules for application to site elements.• Requires additional detail in description.• A clear illustration of the visual design of the site.	<ul style="list-style-type: none">• Style guide clearly describes how color, typography and stylistic elements are to be implemented & applied to the site.• An excellent illustration of the visual design of the site - showing the application of the style guide to key elements using representative content.
HTML Usage	<ul style="list-style-type: none">• HTML is semantically appropriate• Some errors (semantic and/or syntactic) remain in the code.	<ul style="list-style-type: none">• HTML is semantically appropriate and syntactically correct.• Minor syntax errors remain in the code.	<ul style="list-style-type: none">• HTML is implemented with a high standard, well-structured and semantically appropriate.• No errors in the code.
CSS Usage	<ul style="list-style-type: none">• Use CSS to style text, headings, links and block elements appropriately.• Some CSS scripts do not work.	<ul style="list-style-type: none">• CSS use is considered.• Styles are provided for text, headings, links, and block elements.• Minor errors remain in the code.	<ul style="list-style-type: none">• CSS is implemented with a high standard• No errors remained
Navigation	<ul style="list-style-type: none">• A basic navigation system using links• No broken links• Any pages can be accessed	<ul style="list-style-type: none">• Navigation is achieved with simple menus and interface implemented with JavaScript.• Interactions with the interface elements are static or just hyperlinked.	<ul style="list-style-type: none">• Navigation is achieved with menus and interface elements with JavaScript.• Elements provide some interaction such as hints, validation, and autocomplete or drive some functionality.

Note that, if you are unsure whether your elective features can be considered as basic or advanced ones, feel free to contact tutors for confirmation.