

COMP3322A Modern Technologies on World Wide Web

Project - Question Discussion Forum

Total 20 points

Overview:

In this project, you are going to design and implement a Question Discussion Forum system where students can post and answer questions. In this project, you will implement the website using both client-side and server-side technologies. You are supposed to build a database for storing user data, questions, and corresponding answers. You will be given the sample design of the database; however, you can design your database schema if you are interested.

Objectives:

1. A learning activity to support ILO 2.
2. The goals of this programming project are:
 - to get a solid experience in full-stack development;
 - to design and implement a web-based Question-Discussion Forum.
 - to apply what you have learned in the course to build a real-life application.

Specifications:

This project consists of two parts:

1. Part 1: Implement the functionalities of the system, which mainly contains the following pages: the main page showing all questions, the login page, the register page, the question detail page, and the question publishing page.
2. Part 2: Design the layout of the web pages by using CSS and responsive web design.

Part 1 (16 points):

Before developing the website, you are supposed to prepare the database. The suggested database schemas are as follows. You are assumed to have the information of at least three users and three questions (and the corresponding answers) in the database. Please visit the Appendix for some samples provided by us.

Table 1: User

Key	Type	Example
uid	String	"uid1"
name	String	"Andy"
email	String	"andy@test.hk"
password	String	"pwd"

The 'uid' is the identification of the user. The 'email' is used to log in the system, so it needs to be exclusive.

Table 2: Question

Key	Type	Example
qid	String	"qid1"
space	String	"JavaScript"

title	String	"How to learn web developing"
content	String	"Where can I find the resources"
answer	[String]	["aid1", "aid2"]
up	[String]	["uid2", "uid3"]
time	String	"21-09-2020"
creatorid	String	"uid1"
creatorName	String	"Andy"

The 'qid' is the identification of this question. The 'space' describes the field of this question. The 'time' indicates the date when the question is posted, while the format of time can be decided by yourself. The 'up' field is an array of uids representing the users who have upvoted this question. The 'answer' field is an array of answer ids representing the answers to this question. The 'creatorid' field holds the uid of the creator while the 'creatorName' field holds the name of the creator.

Table 3: Answer

Key	Type	Example
aid	String	"aid1"
qid	String	"qid1"
content	String	"You can find some projects in the GitHub."
uid	String	"uid2"
uname	String	"Bob"

The 'aid' is the identification of this answer. The 'qid' shows that this answer is for the question identified by "qid1". The 'content' field is for the content of this answer. The 'uid' and 'uname' show the identity and name of the answerer.

Section 1: Create Five Pages (5.7 points)

1. Main Page (2.7 points)

These are the expected features of the main page when the user has not logged in (shown in Figure 1):

- Top Bar (0.5 points):
 - Name of this system (decided by yourself)
 - "Home" Button, "Hot" Button
 - Search box
 - "Log in" Button: Direct to the login page which will be introduced later.
 - "Register" Button: Direct to the registration page which will be introduced later.
- Left Bar (0.2 points):
 - Four spaces: "Algorithm", "Machine Learning", "System", "JavaScript"
- Question List: List all questions; each question box contains
 - (1 point) Space, Creator Name, Creation Time, Title, Question Content
 - (0.25 points) A upvote button on the left bottom corner of the question box
 - (0.25 points) An answer button on the right of the upvote button

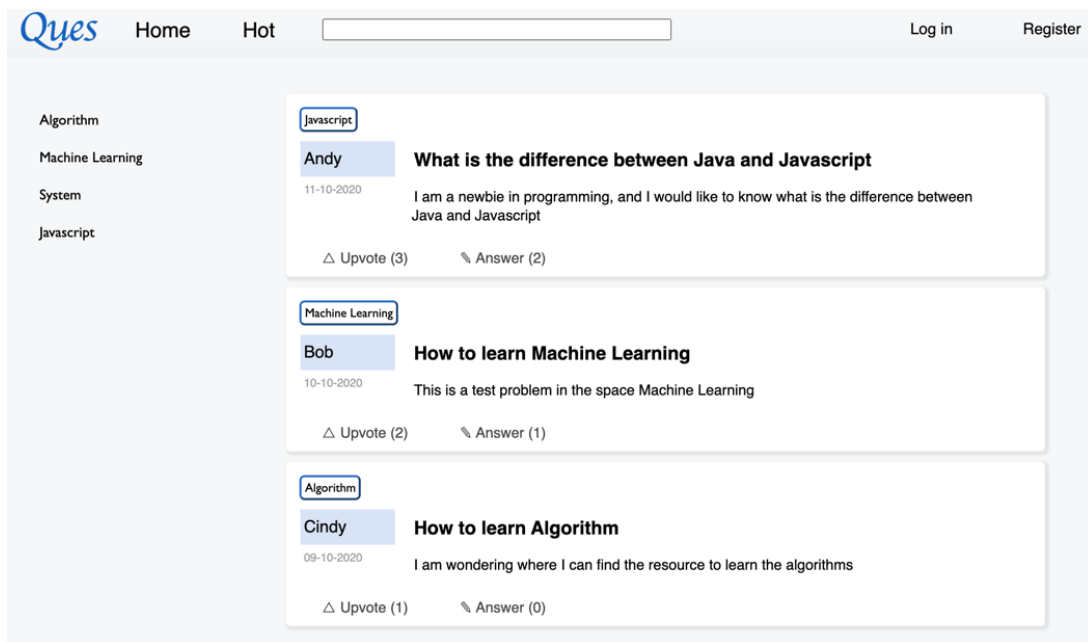


Figure 1: Sample of the main page when the user does not log in (you only need to implement the basic skeleton at the current stage)

You are expected to change the main page when the user has logged in (shown as figure 2).

- (0.1 points) Hide the "Register" button
- (0.1 points) Change the "Log in" button to "Log out"
- (0.1 points) Show an "Ask Question" button on the right
- (0.2 points) A box on the top of question boxes, contains
 - * the name of the user
 - * a label shows "What is your question?"

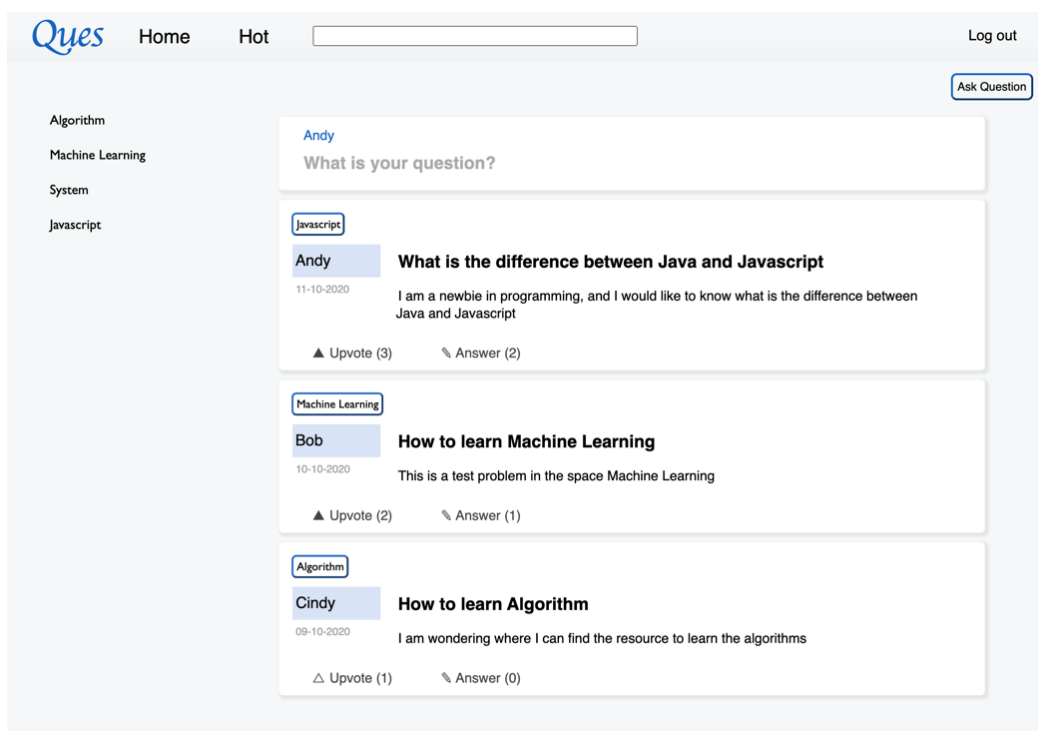


Figure 2: Sample of the main page when Andy logged in the system

(you only need to implement the basic skeleton in the current stage)

2. Login Page (0.4 points)

In the Login Page, you are expected to implement a login form that uses the "POST" method to send the data that is entered by the user in the email and password textboxes. The items are as follows:

- "Email" textbox: Allows the user to enter the email account.
- "Password" textbox: Allows the user to enter the password.
 - The password should not be visible to others when the user is entering the password.
- "Log in" button: Send the form data to the server once it is clicked.
- "Register" button: Direct to the registration page when clicked.

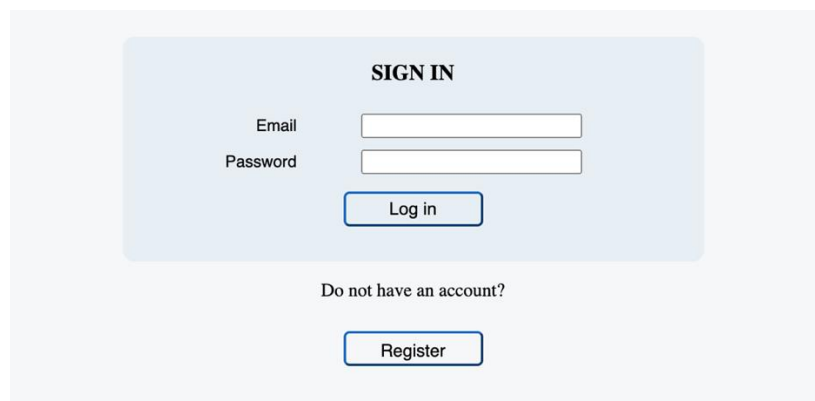
A sample login page with a light blue background. At the top, the text "SIGN IN" is centered. Below it, there are two text input fields: "Email" and "Password". The "Password" field has a small eye icon to its right, indicating a toggle for visibility. Below the input fields is a "Log in" button. At the bottom, there is a link "Do not have an account?" and a "Register" button.

Figure 3: Sample of the login page

(you only need to implement the basic skeleton in this stage)

3. Register Page (0.5 points)

On the Register page, you are expected to implement a registration form that uses the "POST" method to send the data that is entered by the user in the textboxes. The items are as follows:

- "Name" textbox: Allows the user to enter the name.
- "Email" textbox: Allows the user to enter the email account.
- "Password" textbox: Allows the user to enter the password.
 - The password should not be visible to users when the user is entering the password.
- "Confirmation" textbox: Allows the user to re-enter the password.
 - It is not visible to users as well.
- "Register" button: Send the form data to the server once it is clicked.

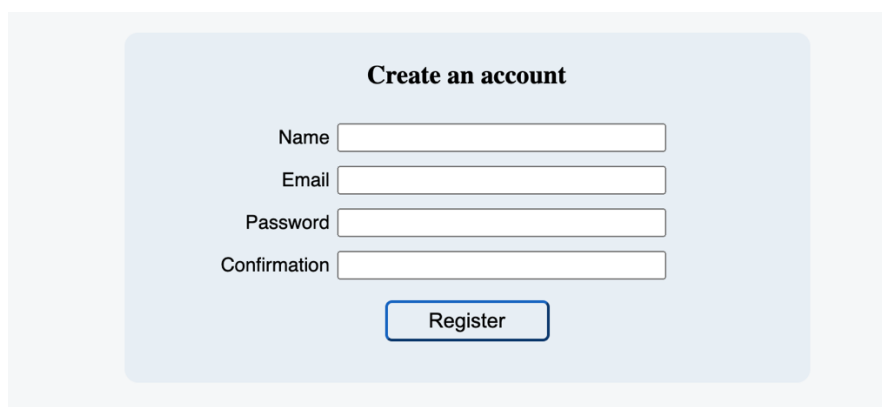
A sample register page with a light blue background. At the top, the text "Create an account" is centered. Below it, there are four text input fields: "Name", "Email", "Password", and "Confirmation". The "Password" field has a small eye icon to its right, indicating a toggle for visibility. Below the input fields is a "Register" button.

Figure 4: Sample of the Register page

(you only need to implement the basic skeleton in this stage)

4. Question Detail Page(1.5 points)

The items are expected to be as follows.

- When use has not logged in the system:
 - (0.1 points) Back Button: Direct to the main page.
 - (0.3 points) Question Detail box that contains
 - title, space, content
 - creator name, time
 - upvote button
 - (0.3 points) Answer boxes: each contains creator name, time, and answer content.

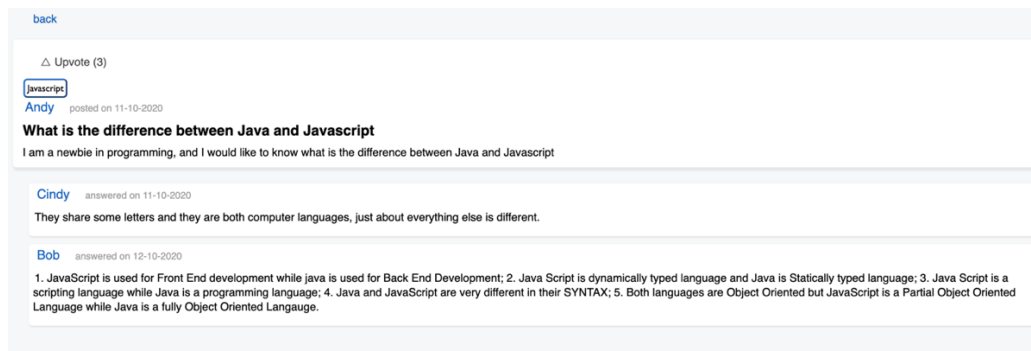


Figure 5: Sample of the Question Detail page when the user does not log in (you only need to implement the basic skeleton in this stage)

- When the user has logged in to the system:
 - (0.4 points) If it is a question created by the user, shows the "Edit" and "Delete" buttons (as shown in Figure 6). If not, do not show the "Edit" and "Delete" buttons.
 - (0.2 points) If this user created some answer(s), show the "delete" button in the answer box(es) (as shown in Figure 7). If not, do not show the "Delete" button.
 - (0.2 points) Another box below all answer boxes that allows user to create a new answer.
 - Shows the name of the user
 - A label shows "Post your new answer."

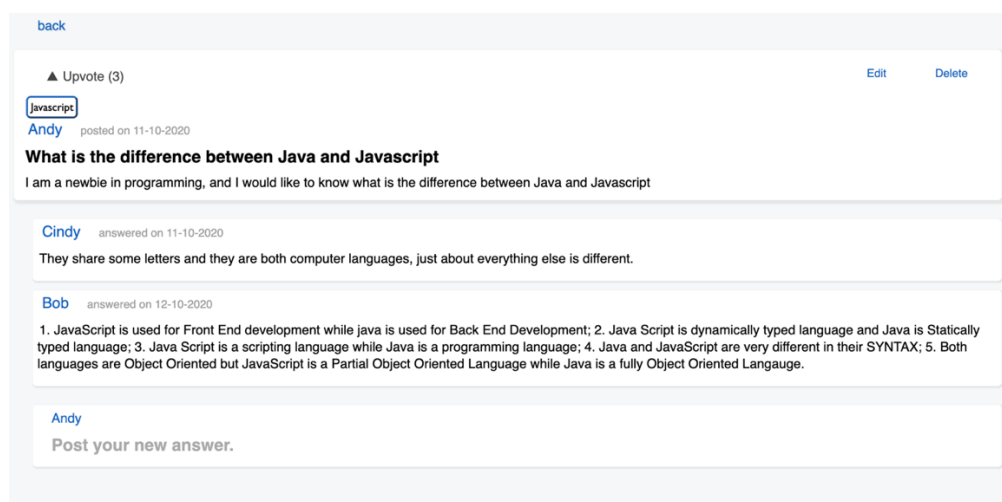


Figure 6: Sample of the Question Detail page when Andy logged in: Question created by Andy (you only need to implement the basic skeleton in this stage)

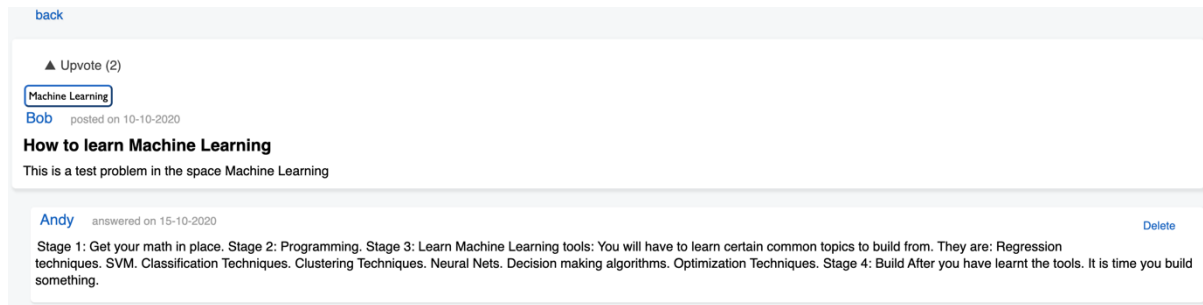


Figure 7: Sample of the Question Detail page when Andy logged in: Question created by Bob (you only need to implement the basic skeleton in this stage)

5. New Question Page (0.6 points)

The items are expected to be as follows.

- (0.1 points) Back Button: Direct to the main page.
- (0.1 points) Title Textbox: Allows the user to enter the title of the question.
- (0.2 points) Space radio input: Allows the user to choose the space.
- (0.1 points) Content Textbox: Allows the user to enter the content of the question.
- (0.1 points) Submit button: Send form data to the server once it is clicked.

Figure 8: Sample of the new question page (you only need to implement the basic skeleton in this stage)

Section 2: Functions (10.3 points)

1. Main Page (5.7 points)

The functions are expected as follows.

- (0.2 points) List all questions in the database.
- (0.2 points) Sort questions by the date by default. The newest question is on the top.
- (0.2 points) Show the number of upvotes and answers near the "upvote" and "answer" buttons.
- (0.2 points) When click the "Hot" button in the top bar, sort the question by the number of votes. If the questions are filtered by space or words, then sort the filtered question only.

- (0.2 points) When click the "Home" button or System Name label ("Ques" at the top left for our example), list all questions that sorted by the date.
- (0.2 points) When click the "Ask Question" button or the "What is your question" label: direct to the New Question Page.
- Filter and search:
 - (0.2 points) When click one of the spaces on the left bar, filter the questions by the space as shown in Figure 9.
 - (0.2 points) Search the question title through words in the search textbox (case insensitive), as shown in Figure 10.

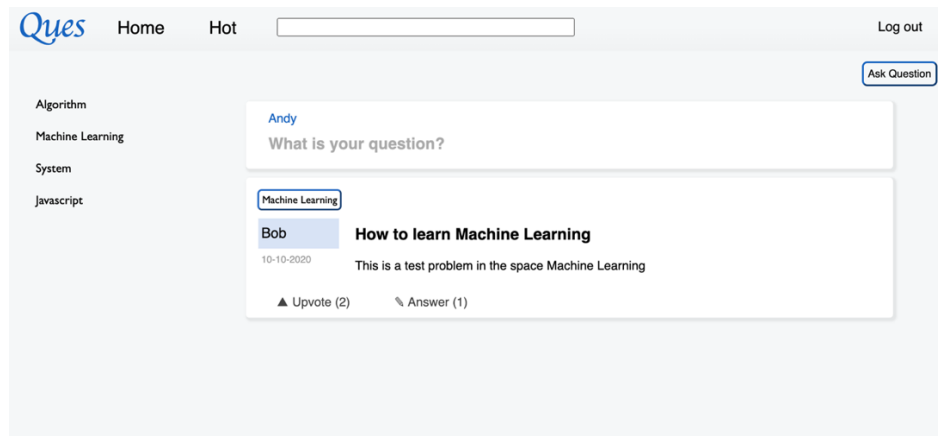


Figure 9: Sample of the space filter.

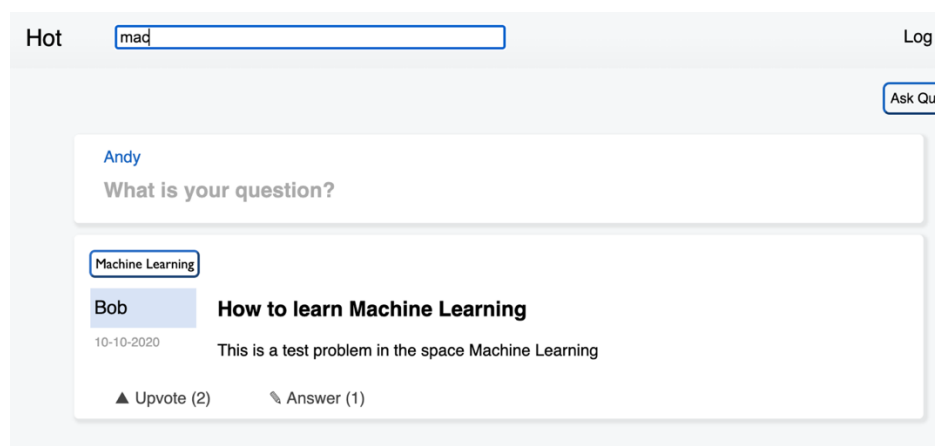


Figure 10: Sample of the title word filter.

- Answer:
 - (0.2 points) When click the title of the question:
 - direct to the Question Detail Page.
 - When click the "Answer" button in the question box,
 - (1 point) list all the answers to this question and
 - (1 point) a new answer box where the user can answer the question directly as shown in Figure 11.
 - (0.5 points) if click the "Answer" button again or click the "Answer" button of another question box, close those answer boxes of this question (and show the answers to the other question).

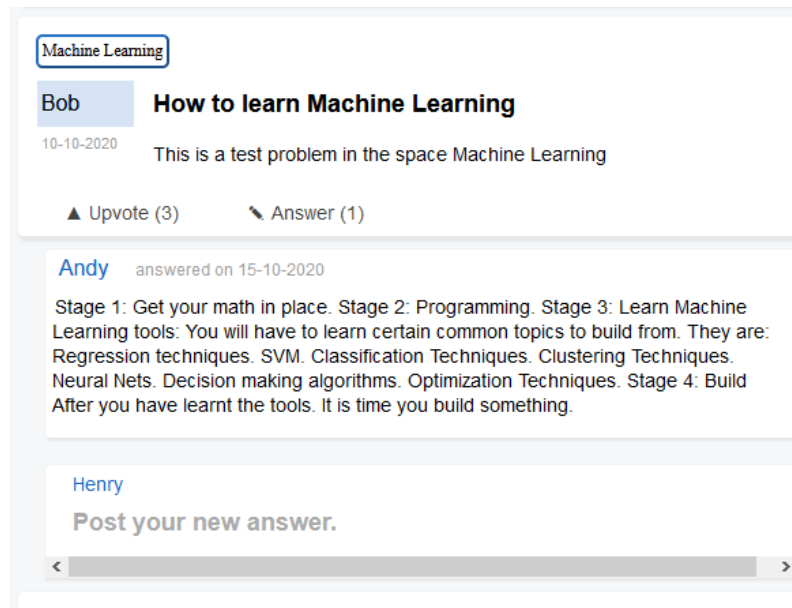


Figure 11: Show the answers below the question box by clicking on the “Answer” button.

- Upvote:
 - (0.2 points) When click the "Upvote" button,
 - upvote this question if the user has not upvoted it before.
 - this cancels the user's upvote state on this question if the user has already voted it.
 - (0.2 points) When the user has voted the question, show a different sign (e.g., solid upvote sign) or text (e.g., "voted") to indicate the upvote state.
- "Show more":
 - (0.5 points) If the content is more than three lines, show only the first three lines with a "show more" button.
 - (0.5 points) If the user clicks the "show more" button, all the content of the question will appear as shown in Figure 12.

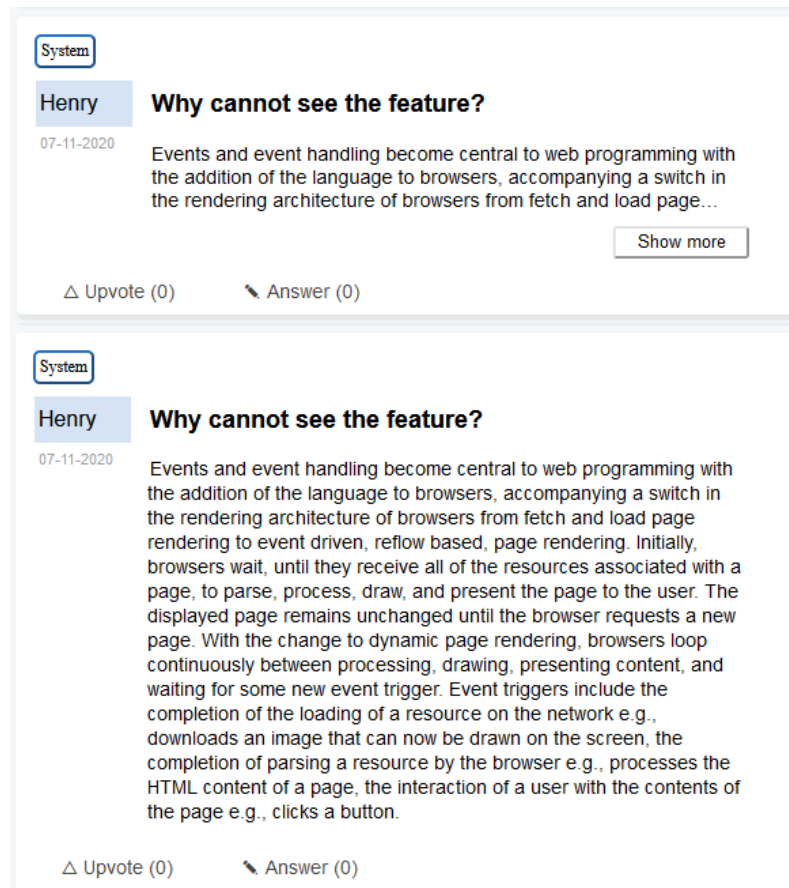


Figure 12: Sample of the “Show more” feature.

2. Login Page (0.5 points)

The functions are expected as follows.

- (0.4 points) If there is an error, alert this message and clear the password.
- Check whether the user has registered in the database; if not, respond "User is not registered".
- Check whether the password is matched; if not, respond "Unauthorized access".
- (0.1 points) If log in successfully, direct to the main page.

3. Register Page (0.6 points)

The functions are expected as follows.

- (0.2 points) Submit the information to the server when the "register" button is clicked; if the user registers successfully, direct to the main page.
- (0.1 points) Check whether every textbox is filled.
- (0.1 points) Check whether the confirmation password is the same as the password.
- (0.1 points) Check whether the email conforms to the standard email format "xx@xx.xx".
- (0.1 points) Check whether the email is in the database; if so, respond "Duplicated user's email address".

4. Question detail page (3.2 points)

Question box:

- Upvote question (similar to the upvote operation in the question boxes on the main page):
 - (0.2 points) When click the "Upvote" button,
 - user votes for this question if the user has not upvoted it before.

- user cancels the upvote state of this question if the user has already voted it.
- (0.2 points) When the user has upvoted the question, show a different sign or text.
- Delete question:
 - (0.2 points) Delete the question and corresponding answers.
 - (0.1 points) If the event is deleted successfully, direct to the main page.
- Edit question:
 - (0.4 points) When click the "Edit" button, present the corresponding textboxes and "Submit" button as shown in Figure 13.
 - (0.4 points) Show the original title, space, and content in the corresponding input boxes.
 - (0.1 points) Check whether every textbox is filled.
 - (0.2 points) Click the "Submit" button, save the edited question.

Figure 13: Sample of edit the question

Answer boxes:

- (0.2 points) Sort the answers by the date. The newest answer is on the bottom.
- Delete Answer:
 - (0.2 points) Delete the corresponding answer.
- Add New Answer:
 - (0.6 points) When click the "Post your new Answer" label, present a textbox, "Submit" and "Cancel" buttons as Figure 14 shows.
 - (0.1 points) Click the "Cancel" button, ignore this answer.
 - (0.1 points) Check whether the textbox is filled before submission.
 - (0.2 points) Click the "Submit" button, post the new answer as Figure 15 shows.

Figure 14: Sample of the new answer box

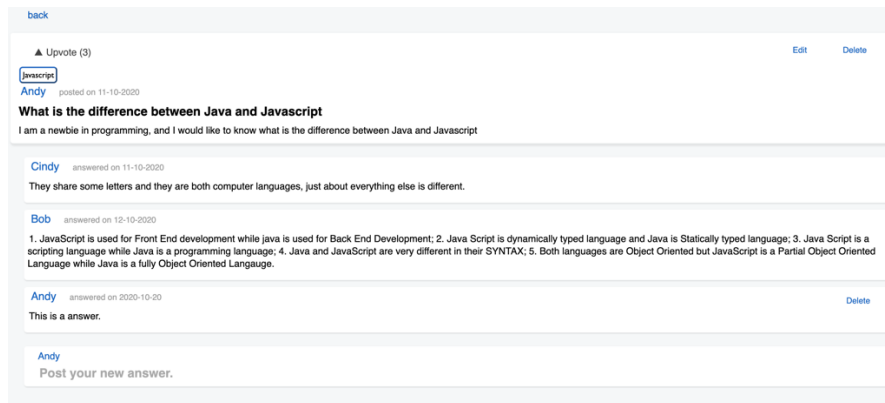


Figure 15: Sample of new answer from Andy.

5. New Question Page (0.3 points)

- (0.1 points) Check whether every textbox is filled.
- (0.2 points) If the new question is published successfully, direct to the main page.

Part 2 (4 points)

In this part, you are going to apply styling with responsive design in all pages that are implemented in part 1. You need to make every element in the pages to be responsive to different mobile/computer screen resolutions (1.5 points).

The remaining 2.5 points are regarding the use of CSS to decorate the pages that you have implemented in Part I. For the CSS implementation, you are free to use any CSS formatting to make the pages look nice. Note: You are not required to have fancy design; just use suitable styling should be fine.

Platform and Technologies

You can use any technologies covered in the course for this project. On client-side, you can use vanilla JavaScript, JQuery, and/or ReactJS to build the front-end of the Website and implement the dynamic features. On the server-side, you can use either PHP+MySQL or Node.js+Express+MongoDB to implement the server-side features.

Submission

Please finish this project on or before **December 2, 2020 (Wednesday)**.

You are required to:

1. Submit your work to the course's Moodle submission page.
2. If you are using Node.js+Express+MongoDB or Node.js+Express+MongoDB+ReactJS to implement the project, here are the instructions to prepare for the submission.
 - i. You must create a project folder "Project" and place all HTML, CSS, JS, and the database 'data' folder in this project folder. Use a compression or archive tool to compress the Project folder and name the file as Project-yourStudentNo.zip (or tgz or other common compression formats). Before creating the project zip file, make sure all source files and data for the MongoDB are all placed inside the project folder. [Recommendation] You can remove the node_modules folder(s) to save space and compression time.
 - ii. Create a text file named "MDBdata.txt" and list all MongoDB statements in creating the database, the collections, and the documents for testing your program. This is

important as the database 'data' folder in the submitted file may be missing or empty. Given these MongoDB statements, tutors can recreate your database to test and grade your project.

- iii. Create a text file named "Readme.txt" that contains:
 - a. List the version numbers of the jQuery, Node.js, Express.js, MongoDB, and ReactJS installed and used on your platform.
 - b. Indicate how to (1) start the MongoDB database, (2) start the Web server (and ReactJS if is appropriate), and (3) access the main page of the Web application.
 - c. Indicate how much work you have completed for the project. You are recommended to make use of the table shown in the Grading Policy section as the checklist. Using it to show us how much you have accomplished.
3. If you are using PHP+MySQL to implement the project, here are the instructions to prepare for the submission.
 - i. You must create a project folder "Project" and place all HTML, CSS, JS, PHP, and supporting files (e.g., images) in this project folder. Use a compression or archive tool to compress the Project folder and name the file as Project-yourStudentNo.zip (or tgz or other common compression formats). It would be nice if you can preserve the directory structure of your files in the archived or compressed file. In that case, by simply decompress or retrieve the files, we can easily set up your Web site for testing and grading.
 - ii. If you are using the Department's MySQL server (Sophia) as your project database server, please make sure that you have included a valid MySQL password in your PHP code so that we can access your database for testing and grading.
 - iii. Create a text file named "DBdata.txt" and list all the MySQL statements in creating the tables and the records for testing your program. This is important if you are using your own MySQL server as the backend database or your password to access Sophia server is missing or not correct. Given these MySQL statements, tutors can recreate your database to test and grade your project.
 - iv. Create a text file named "Readme.txt" that contains:
 - a. Clearly indicate whether you are using the department's MySQL server or your own MySQL server.
 - b. List the version number of the JQuery library if appropriate.
 - c. Indicate how to (1) setup and access the MySQL database and (2) access the main page of the Web application.
 - d. Indicate how much work you have completed for the project. You are recommended to make use of the table shown in the Grading Policy section as the checklist. Using it to show us how much you have accomplished.

Please submit the project.zip file, MDBdata.txt/DBdata.txt file, and the Readme.txt file to the Moodle submission system.

Please note that this project will mainly be **marked by features** and **DO NOT** expected that we will evaluate your source code during marking.

Grading Policy

Part 1 (16 points)	Create pages (5.7 points) - Main Page (2.7 points) - Login Page (0.4 points)
------------------------------	---

	<ul style="list-style-type: none"> - Register Page (0.5 points) - Question Detail Page (1.5 points) - New Question Page (0.6 points) Functions (10.3 points) <ul style="list-style-type: none"> - Main Page (5.7 points) - Login Page (0.5 points) - Register Page (0.6 points) - Question Detail Page (3.2 points) - New Question Page (0.3 points)
Part 2 (4 points)	<ul style="list-style-type: none"> - Responsive web design (1.5 points) - CSS styling design (2.5 points)

Plagiarism

Plagiarism is a very serious academic offence. Students should understand what constitutes plagiarism, the consequences of committing an offence of plagiarism, and how to avoid it. **Please note that we may request you to explain to us how your program is functioning as well as we may also make use of software tools to detect software plagiarism.**

Appendix A: User Information

User 1:

Key	Value
uid	"uid1"
name	"Andy"
email	"andy@test.hk"
password	"pwd"

User 2:

Key	Value
uid	"uid2"
name	"Bob"
email	"bob@test.hk"
password	"pwd"

User 3:

Key	Value
uid	"uid3"
name	"Cindy"
email	"cindy@test.hk"
password	"pwd"

Appendix B: Questions and Answers

Question 1:

Key	Example
qid	"qid1"

space	"Javascript"
title	"What is the difference between Java and Javascript"
content	"I am a newbie in programming, and I would like to know what is the difference between Java and Javascript"
answer	["aid1", "aid2"]
up	["uid2", "uid3"]
time	"11-10-2020"
creatorid	"uid1"
creatorName	"Andy"

Answers to Question 1:

Answer 1:

Key	Example
aid	"aid1"
qid	"qid1"
content	"1. JavaScript is used for Front End development while java is used for Back End Development; 2. Java Script is dynamically typed language and Java is Statically typed language; 3. Java Script is a scripting language while Java is a programming language; 4. Java and JavaScript are very different in their SYNTAX; 5. Both languages are Object Oriented but JavaScript is a Partial Object Oriented Language while Java is a fully Object Oriented Language."
uid	"uid2"
uname	"Bob"
time	'12-10-2020'

Answer 2:

Key	Example
aid	"aid2"
qid	"qid1"
content	"They share some letters and they are both computer languages, just about everything else is different."
uid	"uid3"
uname	"Cindy"
time	'11-10-2020'

Question 2:

Key	Example
qid	"qid2"
space	"Machine Learning"
title	"How to learn Machine Learning"
content	"This is a test problem in the space Machine Learning"
answer	["aid3"]
up	["uid1", "uid3"]
time	"10-10-2020"

creatorid	"uid2"
creatorName	"Bob"

Answer to Question 2:

Key	Example
aid	"aid3"
qid	"qid2"
content	"Stage 1: Get your math in place. Stage 2: Programming. Stage 3: Learn Machine Learning tools: You will have to learn certain common topics to build from. They are: Regression techniques. SVM. Classification Techniques. Clustering Techniques. Neural Nets. Decision making algorithms. Optimization Techniques. Stage 4: Build After you have learnt the tools. It is time you build something."
uid	"uid1"
uname	"Andy"
time	'15-10-2020'

Question 3:

Key	Example
qid	"qid3"
space	"Algorithm"
title	"How to learn Algorithm"
content	"I am wondering where I can find the resource to learn the algorithms"
answer	[]
up	["uid1", "uid2"]
time	"09-10-2020"
creatorid	"uid3"
creatorName	"Cindy"

No Answer to question 3.