Last updated: Feb 25, 2024

Faculty of Information and Communication Technology

ITCS212 Web Programming

Project Phase II (64 pt. + 5 extra pt.)

Assigned Date: Feb 26, 2024

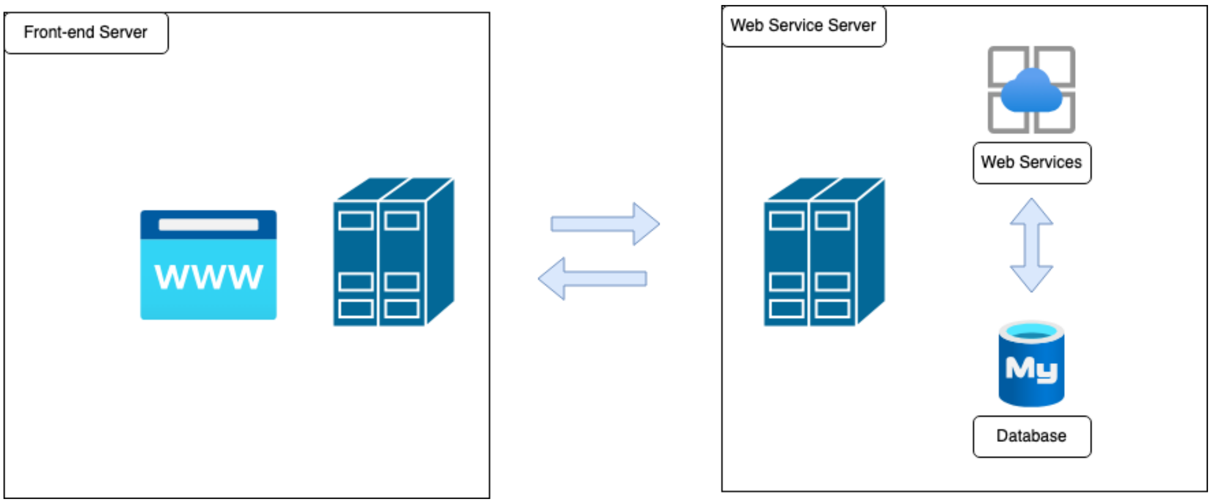
In this team project, you are about to develop a web application for the selected business domain, for example, online shopping, music and entertainment, food services etc.

Project Description

Phase II gives you an opportunity to implement the front-end part and the back-end part of your web application designed in Phase I. You will implement the web application as followed:

- Front-end pages using HTML, CSS, and JavaScript; all pages are hosted in the front-end server.
- Back-end parts including database and web services for your web application using Node.js. All components are hosted in the back-end server
- In addition, you are going to interact with your own web services and a *public* web service using JavaScript.

The simple architecture of this system shown below. With the group of 4 - 5 students, there are 4 tasks to complete:



Task 1: Front-end Implementation

You will implement all the web pages designed in Phase I. Minor changes on design is acceptable. Each page should have the following details:

1. **Home Page:**
    - The page **must** have a navigation bar indicating the sitemap to other pages.
    - The page **must** have a link or button for accessing the Login Page
    - Put some contents (e.g., texts or images) that are meaningful and related to your business domain.
2. **“Login” Page for Administrators (1 page)**
    - The page should have a form consisting of input boxes and buttons for authentication.
    - The page must connect to the homepage/navigation bar.
3. **“Search” Page for searching product/services (1 page)**
    - The page should have a form consisting of a search box and criteria for searching (e.g., search by product name, product type). The search result will be shown on the same page.
    - There must be at least 3 criteria used for searching (exclude No criteria search), for example, search by names, search by rating, search by product/service types.
      Note: searching and filtering are different.
    - The search result will be shown as a creative table or list on the same page.
    - Each product or service of the output should go to its “Detail” page (4)
4. **“Detail” page for the product/service result**
    - The detail page shows the information regarding the selected product/service.
    - The detail page should link to the previous search result from the “Search” page (3)
    - This page should be a template for the details for showing the information fetched from the web services.
5. **“Product/Service Management” page for administrators**
    - The page provides a form to add a new product/service.
    - The page should allow the administrators to modify the existing products/services.
    - The page should allow the administrator to delete the specific products/services.
6. **“User Account Management” page for administrators**
    - The page provides a form to add a new user.

- The page should allow the administrators to modify the existing users.
- The page should allow the administrator to delete the specific users.

7. **“Team” page (1-5 pages)**
   - Have the information in detail about your team members.
   - (*Optional*) Each member can have his/her individual page.

Additional requirements for your web application

- Every page must have access to the navigation bar (i.e., link back to your home page and other pages)
- Each page’s code must be well-structured (indented) and has a reasonable number of comments.
- Each page must use semantic elements (e.g., `<header>` `<footer>` `<article>`)
- Each page must have either **external or internal CSS**. You are welcome to create your own CSS, apply any CSS frameworks, or the ready-made CSS style. Just to make sure you understand the code.
- The front-end part needs to have its own web server.

Task 2: Database Implementation

The database needs to be implemented in DBMS, preferably MySQL. Your database must store:

1. Administrator information (e.g., first name, last name, address, age, and email),
2. Administrator login information (e.g., username, password, role, and login log), and
3. Your business domain data (e.g., product info, services).

Note that you will use your database designed in Phase I. In addition, in your database, you must use **meaningful data** (at least 10 data for each table).

Task 3: Web Services

You are required to create web services to handle the following functions:

1. Authentication web service for administrators. This web service will be called when the administrators login to the web application.
2. Product/Service Search and Details: This web services will be called when any users search and view products or services in the system. The search function supports the following functions:
   - **No criteria search**: the web service will return all results.

- **Criteria search**: The web service must support at least 3 criteria. Therefore, when the users make a search request, they can enter all criteria or some criteria.

3. Product/Service management web service for administrators. The web services allow the administrators to:
   - Insert products or services in the system
   - Update products or services in the system
   - Delete products or services in the system
4. User management web service for administrators. The web services allow the administrators to:
   - Insert administrators' data in the system
   - Update administrators' data in the system
   - Delete administrators' data in the system

In addition, you need to provide at least 2 test cases for each service in the comment above the services. Your service will be tested by **Postman** [ref]. For example,

```
// Testing Insert a new Student
// method: post
// URL: http://localhost:3000/student
// body: raw JSON
// {
//   "student": {
//       "STU_ID": 99998,
//       "STU_FNAME": "Wudhichart",
//       "STU_LNAME": "Sawangphol",
//       "STU_AGE": 35
//   }
// }
```

Task 3: Web Services Interaction

This task aims to connect your front-end part implemented to the web service in Task 3 using JavaScript, e.g., **Fetch** API. In this task, you are required to create connections from the interface to the web services for all functions and pages. In addition, your web application needs to connect to **one public web service** that is useful for your web application, e.g., Google Maps can be used to show the location of stores. Note that the front-end (UI) and back-end (Web service) need to be implemented on different web server.

Task 4: Report

This task requires students to write a report of your work in project phase I and II. The report needs to include:

1. Cover page that contains Faculty name, Unit name, and the information of members
2. Project Overview
3. Navigation Diagram of your web application
4. Details of your web application and code
5. Details of your web services and code
6. The testing results of web services using Postman or any program for testing web service

Additional requirements for your web application and web services

- All web services **must** be implemented using **Node.js or other JavaScript frameworks.**
- All web services **must** be connected to the database server.
- The front-end (UI) and back-end (Web service) need to be implemented on different web server.
- You are required to ensure that your submission can be deployed in the machine of lecturers or LAs.
- The project must have **README.txt** explaining how to run/start your web application and web service. If there are extra remarks, bugs, or exceptions, please state clearly in the readme file.
- The size of the whole project must not exceed 50MB. If you have lots of images, please store the images at the external cloud storage and use the URL for the references.

*** Please read the marking criteria for the project below ***

Instruction for Submission

1. Task 1: You need to submit all front-end source files, encompassing HTML, CSS, and supplementary files (e.g., images), **excluding the 'node_modules' directory**. Rename your source code directory to **`secX_grY_fe_src`**.
2. Task 2: You need to export your database as sql file and rename your sql file to **`secX_grY_database.sql`**
3. Task 3: You need to submit all web service source files, **excluding the 'node_modules' directory**. Rename your source code directory to **`secX_grY_ws_src`**.
4. You need to put all files and folders from (1), (2), and (3) in the directory **`secX_grY_src`** and README file (README.txt) explaining how to run your program.
5. Zip all files from Step 1 to Step 3 into **`secX_grY_pj2_IDxxx_xxx_xxx_xxx.zip`**
6. You need to ensure that your zipped file can be uploaded on Mycourses.

7. Submit the zipped file on MyCourses by the due date:

Tuesday April 19, 2024, at 11:55 PM BKK Time (UTC+7)

(For all sections)

Note: No late submissions will be accepted.

Academic Integrity

Please do your own group work. Your survival in the subsequent courses heavily depends on the programming skills that you harvest in this course. Though students are allowed and encouraged to discuss ideas with others and search for the online sources, the actual solutions must be written by themselves without being dictated or looking at others' code. Collaboration in writing solutions with other groups is not allowed, as it would be unfair to other students. It is better to submit a broken program that is a result of your own effort than taking somebody else's work for your own credit! Students who know how to obtain the solutions are encouraged to help others by guiding them and teaching them the core material needed to complete the project, rather than giving away the solutions.

Front-end Marking Criteria (26 pt.)

All functions in the description need to be preserved. Otherwise, the score will be reduced accordingly.

Criteria	Level 2	Level 1	Level 0
1. **Homepage**	A page has **all** required details	A page has **some** required details	**No** page is implemented
2. **Login page**	A page has **all** required details	A page has **some** required details	**No** page is implemented
3. **Search page**	A page has **all** required details	A page has **some** required details	**No** page is implemented
4. **Detail page**	A page has **all** required details	A page has **some** required details	**No** page is implemented

Criteria	Level 2	Level 1	Level 0
5. **Product/Service Management page**	A page has **all** required details	A page has **some** required details	**No** page is implemented
6. **User Management page**	A page has **all** required details	A page has **some** required details	**No** page is implemented
7. **Team page**	A page has **all** required details	A page has **some** required details	**No** page is implemented
8. **Semantic elements**	The semantic elements are applied to **all pages**	The semantic elements are applied to **some pages**	**No semantic element** is applied at any pages.
9. **Navigation bar**	Have a navigation bar for every page and all pages are linked	Have a navigation bar for every page or all pages are partially linked	**No a navigation bar** is implemented
10. **CSS**	**All pages** are applied the external or internal CSS correctly	**Some** pages are applied the external or internal CSS correctly	**No page** is applied the external or internal CSS (i.e. only use HTML or in-line CSS)
11. **readme.txt**	The steps for running the project are well-written and the project can be run smoothly	Incomplete or missing steps in the readme file	**No readme file or No instruction**
12. **Submission**	**All** submitted files are in the **correct format and named**	**One** of the submitted files is in the **correct format and named**	**No** submitted files are in the correct **format** and **named**
13. **Source code and comments**	**All pages** are appropriately **indented** and **commented**	**Some** pages are appropriately **indented** and **commented**	**No page** is **indented** and **commented**

Web Services and Interaction Marking Criteria (28 pt.)

All functions in the description need to be preserved. Otherwise, the score will be reduced accordingly.

Criteria	Level 2	Level 1	Level 0
1. **Authentication Service**	**Authentication** Service is implemented, and it **works properly**.	**Authentication** Service is implemented, and it is **syntactically correct**.	**No Authentication** Service is implemented.
2. **Search Service**	**Search** Service is implemented, and it **works properly**.	**Search** Service is implemented, and it is **syntactically correct**.	**No Search** Service is implemented.
3. **Insert Service**	**Insert** Service is implemented, and it **works properly**.	**Insert** Service is implemented, and it is **syntactically correct**.	**No Insert** Service is implemented.
4. **Update Service**	**Update** Service is implemented, and it **works properly**.	**Update** Service is implemented, and it is **syntactically correct**.	**No Update** Service is implemented.
5. **Delete Service**	**Delete** Service is implemented, and it **works properly**.	**Delete** Service is implemented, and it is **syntactically correct**.	**No Delete** Service is implemented.
6. **Test cases**	**All** test cases are provided.	**At least 50 %** of test cases are provided.	**No** test cases or less than 50% of test cases are provided**.**
7. **Authentication Service interaction**	**Authentication** Service interaction is implemented, and it **works properly**.	**Authentication** Service interaction is implemented, and it is **syntactically correct**.	**No Authentication** Service interaction is implemented
8. **Search and View Service interaction**	**Search and View** Service interaction is implemented, and it **works properly**.	**Search and View** Service interaction is implemented, and it is **syntactically correct**.	**No Search and View** Service interaction is implemented

9. **Insert Service interaction**	**Insert** Service interaction is implemented, and it **works properly**.	**Insert** Service interaction is implemented, and it is **syntactically correct**.	**No Insert** Service interaction is implemented
10. **Update Service interaction**	**Update** Service interaction is implemented, and it **works properly**.	**Update** Service interaction is implemented, and it is **syntactically correct**.	**No Update** Service interaction is implemented
11. **Delete Service interaction**	**Delete** Service interaction is implemented, and it **works properly**.	**Delete** Service interaction is implemented, and it is **syntactically correct**.	**No Delete** Service interaction is implemented
12. **Public Web Service interaction**	**Public Web** Service interaction is implemented, and it **works properly**.	**Public Web** Service interaction is implemented, and it is **syntactically correct**.	**No Public Web** Service interaction is implemented
13. **Submission**	**All** submitted files are in the **correct format and named**	**Some** of the submitted files is in the **correct format and named**	**No** submitted file is in the correct **format** and **named**
14. **Source code**	All necessary blocks of code have comments.	Some necessary blocks of code have comments.	There is no comment.

Note:

- "Work properly" means when we test the web services, they should return results correctly.
- "Syntactically correct" means all codes for each web service are written correctly, but it cannot be tested, or it does not return any result.

Report Marking Criteria (10 pt.)

Criteria	Level 2	Level 1	Level 0
1. Title Page	All required information is well-written.	Most of the required information is well-written.	Missing information.

2. Organization	Proper formatting, sections clearly labeled, well-organized, professional style	Some formatting errors or missing sections	Multiple formatting errors
3. Topics	All the requested topics are covered in good detail.	Some requested topics absent or details lacking.	Multiple topics lacking or insufficient details.
4. Grammar, Usage, Mechanics, Spelling	No errors	Only one or two errors	More than two errors
5. Quality of information	Interesting and informative.	Some details vague	Details somewhat sketchy

Extra Task (5 pt.)

1. To secure your web services, you are required to use JSON Web Tokens to help you to authenticate the web services (2 pt.).
2. The extra point will be given if your web services are deployed on the provided server (3 pt.).

To get this extra score, you need to finish all requirements above first. In addition, the extra task needs to be express clear in the report and README.txt.