Phase 2B: Data Presentation & Management (Deadline: 1 March 2026) **Same as Phase 3** (Subtotal: 18')

In this phase, you will implement the core functions of the website with mainly Node and SQL.

1. SQL: Create a database with the following structures ______/ 1'
   - o A table for *categories*
     - ▪ Required columns: *catid (primary key), name*
     - ▪ Data: at least 2 categories of your choice
   - o A table for *products*
     - ▪ Required columns: *pid (primary key), catid, name, price, description*
     - ▪ Data: at least 2 products for each category
2. HTML, Node** & SQL: Create an *admin panel* *[+Backend functions]*
   - o Design several HTML forms to manage* *products* in DB, required features: ______/ 2'
     - ▪ Dropdown menu to select *catid* according to its *name*
     - ▪ Input fields for inputting *name*, *price*
     - ▪ Textarea for inputting *description*
     - ▪ ^ File field for uploading a product image (format: at least 1 image format, size: <=10MB)
   - o Design several HTML forms to manage* *categories* in DB ______/ 2'
   - o Submitting the Form to the backend server API results in a DB update. ______/ 3'
     - ▪ (part of Phase 4 Requirement) Try to apply input validation and sanitization

* In terms of management, it includes the capabilities of insert, update, and delete products

^ For the file uploaded, store it at your backend server with its name based on the product ID. You may also store related image paths in the database for the product.

______/ 1'

3. HTML, Node**, SQL: Update the *main page* created in Phase 1
   - o Populate the *category list* from DB ______/ 1'
     - ▪ It can be server-rendered or updated on client-side with JavaScript
   - o Based on the category picked by user, populate the corresponding *product list* from DB ______/ 3'
     - ▪ e.g., the *catid=[x]* is reflected as a query string in the URL (or other method)
4. HTML, Node** & SQL: Update the *product details page* created in Phase 1 ______/ 2'
   - o Display the details of a product according to its DB record
5. Supporting automatic image resizing for product images ________/3'
   - o When a large image is uploaded, the server will resize it (to a fixed, reasonable resolution) and show a thumbnail image. [e.g., two image files with different names for a product]
   - o On the main page, display thumbnails. In the product description page, display the larger image.

**: Other backend languages accepted

Submit the zipped source code to the Blackboard submission box.

- The general principle is that (you can) recreate the frontend/backend from your submitted code with ease. **You shall remove/mask all secret information, e.g., API tokens, secret keys, passwords, etc**.
- For Nginx/Apache/other web server: submit your config files.
- For Node server: submit package.json with your server logic codes (no need to submit installed node modules)
- For Database, SQLite: you can submit the DB file or a backup script to recreate the database. For other databases, submit the script that could rebuild the database (with instructions).

Phase 3: AJAX Shopping List (Deadline: 1 March 2026) **Subjected to change** (Subtotal: 10')

In this phase, you will implement the shopping list, which allows users to shop around your products. This phase is designed to let you practice JavaScript programming.

1. JS: Dynamically update# the *shopping list*
   - When the *addToCart* button of a product is clicked, add it to the shopping list ______/ 1'
     - The addToCart button should function in both main/product page
     - Adding the same product twice will display only one row of record
   - Once a product is added,
     - Users are allowed to update its *quantity* and delete it with a number input, or ______/ 1'
       two buttons for increment and decrement or other reasonable design
     - Store its *pid* and *quantity (you can store more)* in the browser's localStorage ______/ 2'
     - Get the *name* and *price* from your backend server (e.g., with *pid* as query) ______/ 3'
     - Calculate and display the total amount at the client-side ______/ 1'
   - Once the page is reloaded, the *shopping list* is restored (no login required) ______/ 2'
     - Page reloads when users browse another category or visit the product detail page
     - Populate and retrieve the stored products from the **localStorage**
   - [Optional] Try to adopt an OOP design for the shopping cart (and cart item).

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