PHP Project: Project Documentation

DE Online Grocery

# 1. Project description (3-5 pages)

Leader: Manj, Dilraj 300331319

Member: Edison CHAN 300370372

Distribution of work

1. Criteria Grading The web application produces errors or warning messages. The output of var\_dump or print\_r are visible on the web app. The web app creates folders and files automatically. -16 **(QA Both of us)**
2. Project submitted and named properly with all assets to Blackboard by Team Leader, file is named according to the naming convention 1 point **(Dil)**
3. Project Description, installation manual, Class Diagram, Meeting Minutes, all completed with relevant details. Class Diagram **(Dil)** is accurate and includes all properties and cardinality. User Manual – Concise, to the point, graphical, labelled properly. 3 points **(Ed)**
4. Good program structure is used, all functions were described with comments, and comments are used where applicable. Naming conventions are followed. 2 points

**(Both of us) - list of function and class with comments**

1. The database was designed following the requirements. The tables/entities were sound and were designed to reflect real life web application. The complexity of the database should be similar to or greater than any database design used in the class practice. The SQL script was included in the project submission. 3 points **(Dil)**
2. CRUD operations are implemented using PDO, DAO and prepared statements. The errors are captured and logged. The user must be given positive confirmation of a CRUD operation. Statistics are coded using SQL queries and displayed. 5 points **(Dil)**
3. All input were validated according to the expected input.The user must be prompted for corrections and stack traces must be logged to a file. 2 points **(Ed)**
4. HTML Forms were coded properly. The layout is efficient and easy to use, and interface controls are intuitive. 2 points **(Ed)**
5. The overall quality and complexity of the submitted project should be greater than any class demo practices and assignments 2 points **(Both of us)**
6. Bonus requirements 2 points **(Ed → Publish on Github)**

Table: user

Fields: This table contains information about the users of the website. Fields might include userid, username, email, password, registration\_date, last\_login, and other user-specific attributes.

Purpose: The "user" table serves as the central repository for all user-related data, handling user authentication, registration, and any user-specific functionalities on the website.

Efficiency: Using a dedicated "user" table ensures that all user-related data is logically organized in one place. This simplifies querying and updating user information, reducing the complexity of the database schema.

Table: product

Fields: This table contains information about the products available on the website. Common fields might include product\_id, name, description, price, category, availability, and other product-specific attributes.

Purpose: The "product" table stores data related to all the products offered on the website. This includes their details, availability, and pricing information.

Efficiency: By having a dedicated "product" table, all the product data is centralized, making it easier to manage and manipulate product information. This separation of concerns helps maintain data integrity and simplifies queries related to products.

Advantages of Having Only Two Tables:

Simplicity: A smaller number of tables means a simpler database schema. It reduces the complexity of the database design and eases the process of development and maintenance. Fewer tables also lead to simpler queries and more straightforward data management.

Data Integrity: Having dedicated tables for users and products ensures data integrity and reduces the chances of redundant or inconsistent data. It provides a clear structure for storing specific information without mixing unrelated data in a single table.

Performance: With fewer tables, database operations tend to be faster, especially when dealing with complex queries involving joins or aggregations. This is because there are fewer data dependencies to resolve, resulting in quicker response times.

Scalability: A simpler database schema with fewer tables can be more scalable. As the website grows and the amount of data increases, it becomes easier to manage and optimize database performance with only two tables.

Ease of Maintenance: Maintenance becomes more manageable as there are fewer tables to update or modify. Any changes to the database structure, such as adding new fields or altering existing ones, can be done more efficiently.

Code Maintainability: A simplified database schema corresponds to simpler and more maintainable code in the application layer. It reduces the complexity of database interactions in the codebase, leading to easier debugging and troubleshooting.

Reduced Redundancy: With only two tables, you avoid redundant data that may occur when multiple tables store similar or overlapping information. This minimizes the risk of inconsistency and saves storage space.

However, it's important to note that while using just two tables might be efficient for smaller projects or specific use cases, larger or more complex websites might require additional tables to accommodate more intricate relationships or functionalities. The decision to keep the database simple or to expand it depends on the specific requirements of the website and the future scalability needs.

- Include a list of features you implement and why these add value to your project.

Certainly! Let's go through each of the added features and explain how they add value to your project:

Search bar:

Value: The search bar enhances the user experience by enabling users to quickly find specific products they are interested in. Instead of manually browsing through pages of products, users can simply enter keywords or product names in the search bar to get relevant results.

Efficiency: With a well-implemented search functionality, users can easily narrow down their choices, leading to increased user satisfaction and retention. Moreover, a search feature streamlines navigation and reduces the time it takes for users to find what they are looking for, ultimately improving the overall website's efficiency.

Item view (with placeholder for pictures):

Value: The item view provides users with detailed information about a specific product when they click on it. This feature allows users to make informed decisions about whether to purchase the product or not.

Visual Representation: While placeholder images may not show the actual product pictures, they still serve an essential purpose by visually representing the products. This enhances the user experience and makes the website more appealing, even if the actual product images are not available or not yet uploaded.

Login system with interactive feedback:

Value: The login system adds a layer of personalization and security to the website. It allows users to create accounts, sign in, and access personalized features such as their order history, wishlists, and preferences.

Interactive Feedback: By providing interactive feedback during the login process (e.g., displaying error messages for incorrect credentials or successful login messages), users are kept informed about the status of their actions. This improves user satisfaction and helps prevent confusion during the login process.

Admin management to add/remove products:

Value: The admin management system empowers the website administrators to have control over the products listed on the website. They can easily add new products, update existing ones, or remove outdated or out-of-stock items.

Efficiency and Flexibility: With admin management, the website becomes more dynamic and adaptable to changing product offerings. It streamlines the process of updating the product catalog, ensuring that users always have access to the most accurate and up-to-date information.

Overall, these added features significantly enhance the website's functionality and user experience. Users can quickly find products they are interested in, get detailed information about items, and have a personalized experience through the login system. Meanwhile, the admin management system allows the website to stay current and relevant by providing control over the products displayed. All of these aspects together contribute to making the website more user-friendly, efficient, and appealing to both users and administrators.

- Include an installation manual, how to set up (database, folder structure, web alias, etc). See Appendix 3

- Include a list with specific examples of how the technical concepts were implemented (see requirements)

- The database:

o Except if it is an associative table that breaks the M:N relationship, each table/entity in the database

should have at least five columns

o If you are in a team of two, the number of entities > count($members). One of the entities can be an

associative entity

o If you are in a team of three, the number of entities > count($members). One of the entities must be an associative entity

- Entities must be stored in a Database and accessed using PDO

There are two entities in the database, see more at Database details below.

User Table which contains the various

- The application must support CRUD (Create, Read, Update and Delete) Items using PDO. This must be

implemented for each entity. It does not mean that you should have the same list view, add view and edit view

for each entity. It should be a complete web application, not a data view and entry application.

- The application must have multiple HTML forms and pages.

- The user must be able to search and lookup for records from the database.

- Statistics must be shown for at least one entity.

- All input must be validated, and the appropriate events handled; proper English must be used.

- The app must be easy to use and visually pleasing, professionally designed with effective use of layouts, text,

page and content elements and user input elements.

- Users must be able to login; their credentials must be encrypted. The application must use sessions logging in

and logging out.

- The above requirements are for passing the project. You should try your best to make your project stand out

by integrating new technologies or topic that are not covered in the class.

- Include a list of the technologies implemented as an extension beyond what was covered in the classroom.

## Appendix 1: Class diagram

Specify the entities, the PK and FK and cardinalities. You can use tools like Software Ideas Modeller or any other

UML drawing tools like lucidchart.com to create your model

## Appendix 2: Meeting minutes

For those working in a group, every time the group meets, be sure to record the meeting minutes. Be sure to

include who was present and provide details on:

- Attendance (who is present for your meeting)

- What was done since the last time the group met (per person)?

- What is working well?

- What is not working well?

- What will be done before the next time the team meets?

Course: CSIS3280-001 BACKEND WEB DEVELOPMENT

Logistics:

Links:

Latest meeting note:

<https://docs.google.com/document/d/1CPxe6M_CGIk4EhlHPLl1iZhRxJfs_5yUqt5mI81Nvvc/edit?usp=sharing>

Google Drive:

<https://drive.google.com/drive/folders/1idI_rNAA2-JcdJI2lJoRg3FRLYv7vzqO?usp=sharing>

Github:

<https://github.com/WCEdison/CSIS3280_PHP_Project.git>

Personal info:

Leader: Manj, Dilraj 300331319

Member: Edison CHAN 300370372

1. Reconfirm key design feature

1. Format: Store front
2. Major tech decision
   1. Use Laravel
   2. Use Github to publish webpage
   3. Find Food/Grocery API
   4. Use local database
3. Design direction
   1. Purpose:
      1. Grocery Store
   2. Features:
      1. Login -> As manager
      2. Customer
         1. View
         2. Filter (Basically Sql select, use desc/asec, sort by)
      3. Management (use from last demo)
         1. Add
         2. Edit
         3. Delete
   3. Database
      1. 2 x Unrelated Tables
      2. Admin login
      3. Product
4. Question to ask: How much weight is this project and when is the deadline
   1. If high then we add more feature (Might expand if scope expands)
   2. Otherwise we keep current feature + grocery api + login for admin + database editing for admin

2. Logistics: How to communicate, Github project and publishing

1. Whatsapp to do normal communication
   1. Error
   2. To dos
2. Teams for bigger meeting → On demand only (one more handing it in)
3. Github + Github desktop
4. Use [Visio] for Class Diagram

---------------------------------------------

3. Task to be done

1. Criteria Grading The web application produces errors or warning messages. The output of var\_dump or print\_r are visible on the web app. The web app creates folders and files automatically. -16 **(QA Both of us)**
2. Project submitted and named properly with all assets to Blackboard by Team Leader, file is named according to the naming convention 1 point **(Dil)**
3. Project Description, installation manual, Class Diagram, Meeting Minutes, all completed with relevant details. Class Diagram **(Dil)** is accurate and includes all properties and cardinality. User Manual – Concise, to the point, graphical, labelled properly. 3 points **(Ed)**
4. Good program structure is used, all functions were described with comments, and comments are used where applicable. Naming conventions are followed. 2 points

**(Both of us) - list of function and class with comments**

1. The database was designed following the requirements. The tables/entities were sound and were designed to reflect real life web application. The complexity of the database should be similar to or greater than any database design used in the class practice. The SQL script was included in the project submission. 3 points **(Dil)**
2. CRUD operations are implemented using PDO, DAO and prepared statements. The errors are captured and logged. The user must be given positive confirmation of a CRUD operation. Statistics are coded using SQL queries and displayed. 5 points **(Dil)**
3. All input were validated according to the expected input.The user must be prompted for corrections and stack traces must be logged to a file. 2 points **(Ed)**
4. HTML Forms were coded properly. The layout is efficient and easy to use, and interface controls are intuitive. 2 points **(Ed)**
5. The overall quality and complexity of the submitted project should be greater than any class demo practices and assignments 2 points **(Both of us)**
6. Bonus requirements 2 points **(Ed → Publish on Github)**

4. To do & milestone

1. Set github
2. Find a Grocery/Food API
3. Learning how to publish webpage github
4. Set up a database design
5. Set up database
6. Set up PDO and DAO
7. Set up validation
8. **Worst Case: Switch to CSS + HTML (if laravel doesn’t work) + Switch to React**

4. User manual (1 page)

Provide a concise user manual how to install, use and operate the application.

- Include visuals wherever applicable

- Must cover all the actions a user can perform (CRUD and Search etc...)