Sprint 2 - Design

Group task 22P

Team Name: Team Grotle
Tutorial Class: Tue 12:30 EN310
Tutor: Naveed Ali
Team members:

Group members	**Student ID**
Hamish	103607352
Dilni	103616345
Kamar	103607585
Melanie	103489466
Justin	102589705
Tevy	103139978
Cormac	102581060

Software Component Design

Block Diagram

The following block diagram shows an overview of the GotoGro-MRM web application:

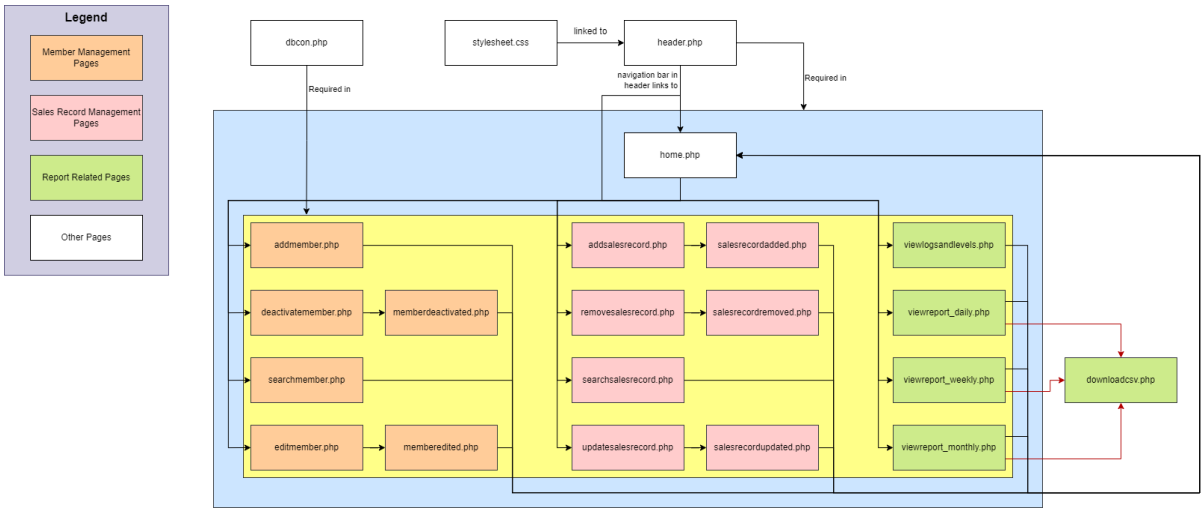

Diagram 1: Block Diagram of GotoGro-MRM after Sprint 2

As shown in the diagram, GotoGro staff will be able to access every function on the system through the home page or using the links provided by the navigation bar in the web application's header. After completing a process, the staff will always be redirected back to the home page.

A header file is included in every PHP file in the blue area, while a database connection file is included in every PHP file in the yellow area.

Architecture Diagram

An architecture diagram has also been made to show the design of the software components for GotoGro-MRM that have been completed in Sprint 2 (see Diagram 4). This diagram is discussed further in the "Design Principles" section under the "Architectural Style" subsection.

Entity-Relationship Diagram (ERD)

A crow's foot notation ERD has been made to depict the relationships between entities in GotoGro-MRM's database.

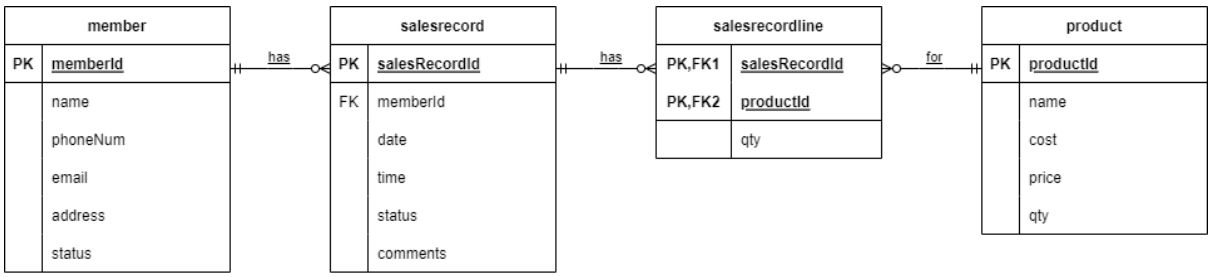

Diagram 2: ERD of GotoGro-MRM

While it is the staff who manages the application's functionalities (such as adding members, adding sales records etc.), they are not an entity in the database. This is because there is no need to record the staff's details in the database, as there is no login system.

Another thing to note is that the reports generated by the application will not be stored as records in the database. This is because there is no reason to do so, seeing that the reports generated are downloadable in comma-separated values (CSV) format. Hence, reports are also not an entity in the database.

Sequence Diagram

In order to showcase the interaction between different software components, we have created a sequence diagram that shows the process of adding a member below:

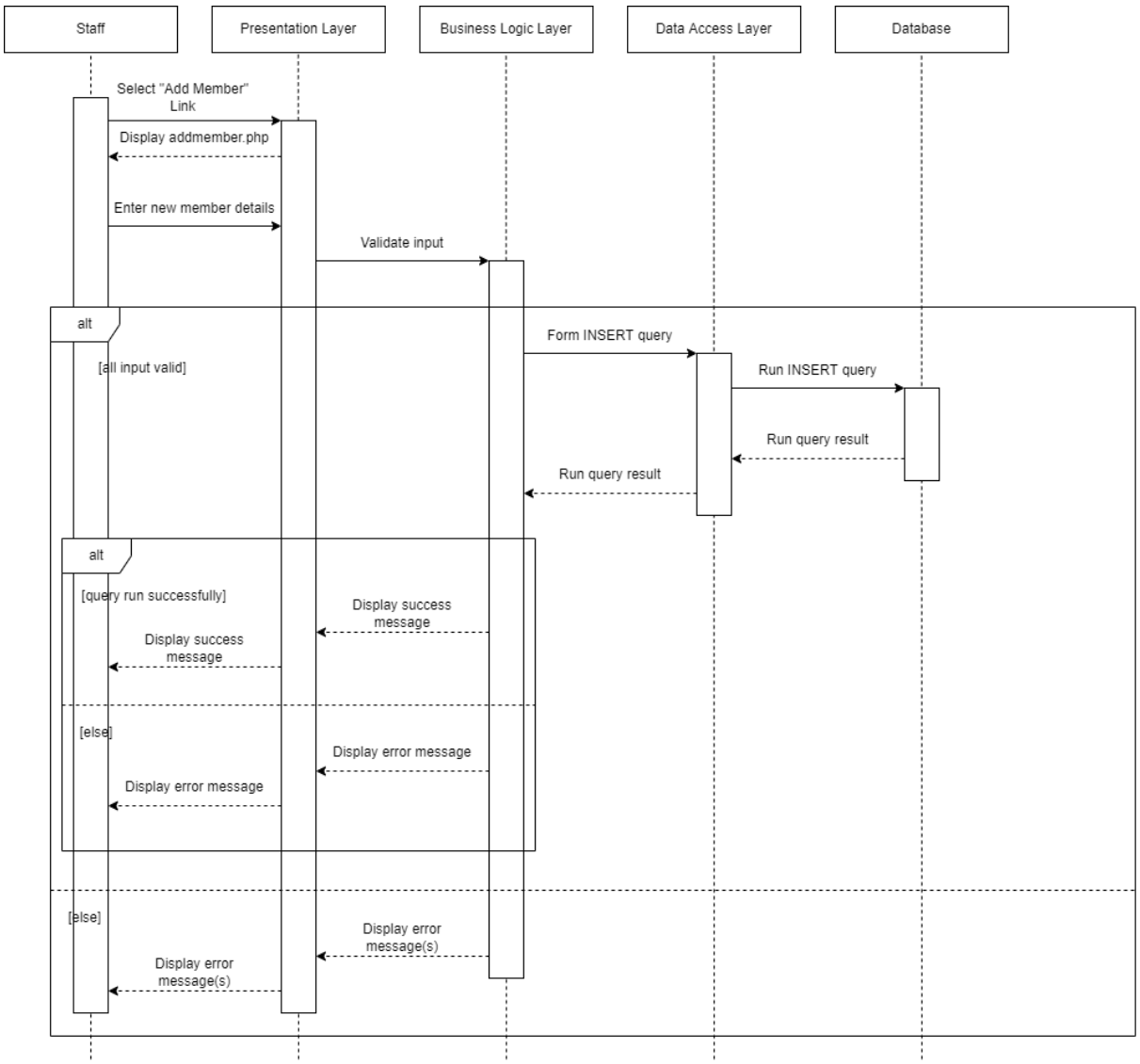

Diagram 3: Sequence Diagram - Add Member

Design Principles

Strong Cohesion and Weak Coupling

Cohesion refers to the degree to which the elements inside a module belong together. We have achieved strong cohesion in this application by binding related methods into their own PHP files. For instance, all the code needed to add a new member can be found in the addmember.php file.

Coupling refers to the level of interconnection between modules. We have achieved weak coupling in this application by separating code that is unrelated to different PHP files. Thus, almost every PHP file can run without relying on code from another file. For example, addmember.php will still be able to run as a standalone page even if the files for other pages (with the exception of base files such as header.php and dbcon.php) are deleted.

Object-Oriented (OO) Principles

We did not implement any OO principles in our application for Sprint 2, since we are unfamiliar with doing so in PHP.

Architectural Style

We have implemented the n-tier architecture in our application for Sprint 2. We have implemented a three-tier architecture specifically, which is composed of a presentation tier, a business tier and a data tier (see diagram 4). The presentation tier displays information to the user, the business tier performs processing, and the data tier stores and retrieves information from the database.

By using this architectural style, we have made a flexible and reusable application. This means that if GotoGro would like to modify or add specific tiers in the future, they can do so without having to rework the entire application.

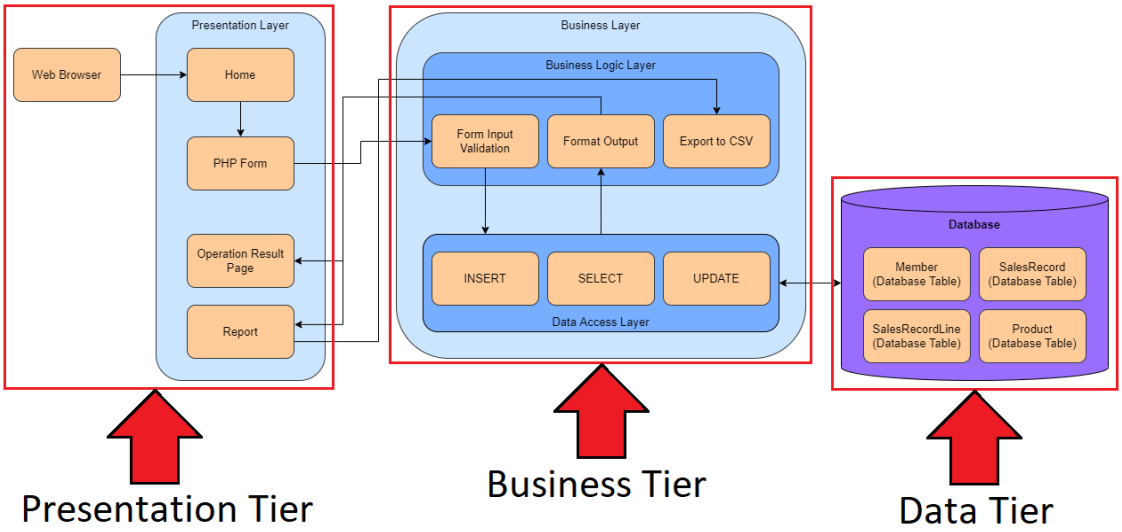

Diagram 4: Architecture Diagram of GotoGro-MRM after Sprint 2