CHAPTER 2

THEORETICAL FOUNDATION

2.1 Theoretical Foundation

In this chapter, all of the existing technologies that is used in the thesis will be mentioned and elaborated. Be advised that although it contains the theories, all the explanations regarding to the thesis itself will not be included in this chapter.

The three sub-chapters (2.2 to 2.4) contains the information regarding the technologies that are used to build this application, which are PHP, XAMPP, and MySQL. The next sub-chapter contains the information regarding all the types of diagrams used as a general explanation which are UML class, entity relationship, data flow and use-case diagram.

2.2 PHP Scripting Language

PHP, which stands for personal home page, is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML **[1]**. Since it is a scripting language, which is easily attached in HTML, developer may use the scripting language and code it even on a plain notepad, and adding a .php extension behind the file name instead of .txt.

It was first created in 1994 as a set of Common Gateway Interface (CGI) binaries written in the C programming language by the Danish/Greenlandic programmer Rasmus Lerdorf**[2][3]**. During 1994, it is named PHP/FI, which stands for

Personal Home Page / Form Interpreter. PHP/FI had Perl-like variables, automatic interpretation of form variables and HTML embedded syntax. The syntax itself was similar to that of Perl, albeit much more limited, simple, and somewhat inconsistent **[4]**. In November 1997, PHP/FI 2.0 is officially released, the second write-up of the C implementation. However, it soon is followed afterwards by PHP 3.0, the very first alpha of PHP language.

```
<!--include /text/header.html-->

<!--getenv HTTP_USER_AGENT-->
<!--ifsubstr $exec_result Mozilla-->
  Hey, you are using Netscape!<p>
<!--endif-->

<!--sql database select * from table where user='$username'-->
<!--ifless $numentries 1-->
  Sorry, that record does not exist<p>
<!--endif exit-->
  Welcome <!--$user-->!<p>
  You have <!--$index:0--> credits left in your account.<p>

<!--include /text/footer.html-->
```

Fig 2.2a – Example of codes of PHP/FI **[4]**

PHP 3.0 was the first version that closely resembles PHP as we know it today. It was created by Andi Gutmans and Zeev Suraski in 1997 as a complete rewrite. One of the biggest strengths of PHP 3.0 was its strong extensibility features. In addition to providing end users with a solid infrastructure for lots of different databases, protocols and APIs, PHP 3.0's extensibility features attracted dozens of developers to join in and submit new extension modules. Arguably, this was

the key to PHP 3.0's tremendous success. Other key features introduced in PHP 3.0 were the object oriented syntax support and the much more powerful and consistent language syntax. The drastic change between the PHP 3.0 and PHP/FI 2.0 causes the PHP to be named differently from its predecessor. Its name is changed to plain PHP, instead of PHP/FI. Sometimes it’s even called by its recursive acronym, which is PHP: Hypertext Preprocessor. In June 1998, PHP 3.0 is officially released after 9 months of beta testing **[4]**.

```
<?php require('../entity/userEntity.php');
session_start();
$command = $_REQUEST['command'];
switch($command){
    case 'addActivity': addActivity(); break;
    case 'updateActivity': updateActivity(); break;
    case 'deleteActivity': deleteActivity(); break;
    case 'doTask': doTask(); break;
    case 'finishTask': finishTask(); break;
}
…
function doTask(){
    $act_id = $_POST['id'];

    $time_now = date('Y-m-d H:i:s');
    $con = mysql_connect("localhost","root","root");
    mysql_select_db("dailyactivities", $con);
    mysql_query("INSERT INTO activitymonitor (start, status,
activity_id) VALUES ('".$time_now."', 'Ongoing', ".$act_id.")")
or die (mysql_error());
    mysql_close($con);
    header('location: ../index.php');
}
?>
```

Fig 2.2b – Code sample of PHP 3.0 onwards

In May 22, 2000**[5]**, PHP 4.0 is released. The PHP 4.0 now supports Zend engine 1.0**[4]** , named after its two creators, which are Zeev and Andi. Not only does it support a new engine, PHP 4.0 also has a great improvement in its performance and including other key features such as its compatibility to many more web servers, enabling HTTP Sessions, and several new language syntax **[4]**.

On July 13, 2004**[5]**, PHP 5.0 is released. This is the latest major version of PHP. The PHP 5.0, powered by the new Zend Engine II **[4]**, includes support for object-oriented programming and PHP Data Objects extension (such as interface for database). A new major version has been under development alongside PHP 5 for several years. This version was originally planned to be released as PHP 6 as a result of its significant changes, which included plans for full Unicode support. However, Unicode support took developers much longer to implement than originally thought, and the decision was made in March 2010**[6]**. to move the project to a branch, with features still under development moved to a trunk. Due to this decision, as of March 4, 2010, the latest PHP version of 5.3.2 is released, containing a few bug fixes and session protection**[7]**.

2.3 XAMPP Apache Friends Web Server

Apache Web Server is a part of a big organization project called “Apache Software Foundation”, which is first found in 1999 **[10]**. It supports many languages of programming including PHP, and licensed for open modification, in which makes it an open source application.

Apache Friends, aside from Apache itself, is a non-profit project to promote the original Apache web server. First founded in the spring 2002 by Kai Oswald Seidler and Kay Vogelgesang **[9]**, Apache Friends are easier to configure for PHP and connecting it to MySQL compared to the real Apache web server. This is the most appropriate web server there is since the technology that this thesis is using is the PHP and MySQL itself.

XAMPP Apache Friends support both MAC and Windows OS. However, there will be not much of an explanation of XAMPP, since it is install and use application, not yet mentioning it is a closed source application.

2.4 MySQL Database

MySQL is a relational database management system (RDBMS) **[11]** that runs as a server providing multi-user access to a number of databases. MySQL code uses C and C++. The SQL parser uses yacc and a home-brewed lexer, sql_lex.cc **[12]**. MySQL works on many different OS, including the two biggest and most commonly used OS, Mac OS X and Microsoft Windows. All major programming languages with a language-specific APIs require libraries to be imported in order to connect and access MySQL database system **[13]**. However, PHP is not considered to be a language-specific API, thus does not require any libraries to be imported, since the web server application (XAMPP, see above section) has already connected the language to the MySQL application through PhpMyAdmin.

Within the Apache server, there are two MySQL API extensions that are compatible to PHP language – namely mysql and mysqli. However, MySQL has not yet been supported until the PHP version was 4.0 and 5.0 (see section 2.2 – PHP Scripting Language) **[16]**.

MySQL was first developed by Michael Widenius and David Axmark beginning in 1994 **[14]**, and was first released internally in 23 May 1995 **[13]**. Up to today, the popularity of MySQL on Windows remains strong due to the fact that MySQL delivers **[15]**:

- Lower TCO
- Ease of use
- Reliability
- Performance
- Fully featured database with no functional limitations

Not only looking at MySQL the application itself, it also has a few storage engines, which may be altered according to the developer's needs. In this thesis however, only one storage engine is used due to the same type of requirement of data access by the users and developers, which is InnoDB storage engine. MySQL is chosen as the best DBMS application is because it suits the most for XAMPP Apache Friends server. There are of course, a few other options than

MySQL. However, to connect it to the web server, some add-ons are required, since MySQL is the only DBMS that XAMPP application comes in bundle with.

2.5 Diagrams

2.5.1 Data Flow Diagram

Data flow diagram (will be referred as DFD) is a graphical representation the flow of data within a system **[20]**. There are many forms of data flow diagram, there are no fixed standard of this diagram. However, the components always stay the same – Process, Data Storage, Interface, and Connector.

Process component represents the functionalities of the applications. It is denoted by a square with soft edges. It is able to receive input data then sending output data, and initiate input data to be sent to another component.

Data storage component represents the processes of dealing with database (queries). It is denoted by a square with a missing right side. Data storage is able to receive input data and reply it with an output data, but not able to initiate the data flow process (sending an input data to other component).

Interface component represents the web page of which the user will be operating on. Be advised that interface is not the same as sitemap, as

interface only display which functionalities can one access when a user has that type of interface; meanwhile sitemap displays which pages are accessible from a page that is currently displayed. Interface component is able to initiate data input and get a response back from a data flow, but not able to receive input initiated by other components.

Connector is the component which connects these three classes. It is denoted by an arrow sign. It also has its captions, describing briefly the data that flows from one component to the other.

2.5.2 Entity Relationship Diagram

Entity Relationship Diagram (will be referred as ERD) is diagram that represents how entities are related to each other. The entities themselves also have attributes defined in them. ERD was first introduced by Charles Bachman, and then later popularized by Dr Peter Chen **[22]**. This is another form of database modeling technique aside from data flow diagram. DFD however, focuses more on the processes of the data flow; meanwhile ERD focuses more on the entities of the database and their relationships.

The entities are the table from the database itself. It quite cannot be called “types” of ERD, but ERD comes in different forms. The simplest ones are the table names only, with their relationships. There is also the most

complex (or complete) one, which contain the table names, its attributes along with its data types, and the relationship types.

There are three types of relationships in ERD. They are **[23]**:

- ✓ One-to-one relationship

  This relationship means that one table can only have one table associated with it. The example of this relationship is a person and a DNA type. One person can only have one type of DNA, and vice versa.

- ✓ One-to-many relationship

  This relationship means that one table can have many tables associated with it, but not the other way around. The example of this relationship is a car and a tire. A car can have many tires, but a tire cannot be put on more than one car.

- ✓ Many-to-many relationship

  This relationship means that one table can have many tables associated with it, and vice versa. The example of this relationship is a course and a student. A course can have many students, and so does a student may have many courses.

2.5.3 UML Class Diagram

Unified Modeling Language (will be referred as UML) class diagram is used for describing the classes and how it will interact to each other **[8]** in this thesis. There are a few standardized symbols and diagram format that

UML class diagram follows. Note that the explanation will only elaborate all the elements used in the UML class diagrams in chapter 4.

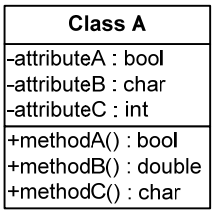

Fig 2.5.3a – UML Class Component

Class diagram consists of a class itself, an entity (if necessary), and a relationship. The diagram above represents a class. A class consists of three segments – class name, attribute name, method / function name. In this explained diagram, all the names are self-representing in order to make easier understanding of the diagram.

The top segment is the class name. In the diagram, it is declared as ‘Class A’. The middle segment is the attribute. An attribute is the variable used within the class, whether it is global or local. The following text after the : sign in the attribute segment represents the data type of each attribute. Different from java which is declares the data type prior to the name, the UML class diagram sets the data type after declaring its name with a : sign as a connector. For instance, attributeA is declared as a boolean, which contain the value of only true or false.

The bottom segment is the method used in the following class. Parameters, however, are ignored in this diagram. That explains why all

the methods have () sign instead of accepting any parameters. Although, the real code may contain parameters. Following the method name, there is a : sign, similar to attribute segment. And just like it, the data type written represents the return value of the following method. For instance, methodB() returns the double data type after executing its method.

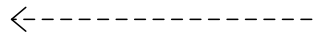

Fig 2.5.3b – Dependency relationship

Dependency is a relationship where one class needs the service provided from another class. It may not be so clear here, but it will be in chapter 4.

Fig 2.5.3c – Association relationship

Association is a relationship where one class can navigate from object of one class to the other objects from another class, usually by following object references **[8]**

2.5.4 Use Case Diagram

The next diagram that will be used in this thesis is the use-case diagram. This diagram consists of two components, which will be elaborated. Again, please be advised that the components explained in this chapter are only the components that will be used within this thesis. Any components that are not related with the thesis are omitted on purpose.

Use-cases

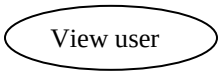

Fig 2.5.4a – Use-case Component

A use-case is defined as a behaviorally related sequence of steps (a scenario), both automated and manual, for the purpose of completing a single business task **[17]**. It is placed on the center of the use-case diagram. It is symbolized as a simple circle, with a text within denoting what is the use case

Actors

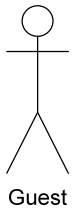

Fig 2.5.4b – Actor Component

Actors are the external users that triggers or initiates use-cases **[17]**. In an application system development, an actor is a user of the system. It is symbolized as a stickman, just like the figure above, and a text is located just below of above the figure, to represent who is the actor.

Association Relationship

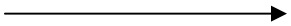

Fig 2.5.4c – Associations Relationship

There are many relationships of use-cases such as associations, extends and depends on. However, only two types of relationship are used in the diagram for the thesis. One of them is association. Association is a relationship between an actor and a use-case exists whenever the use case describes an interaction between them **[17]**. It is symbolized as an arrowhead, like the figure above.

Depends on Relationship

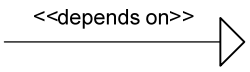

Fig 2.5.4d – Depends on

The other type of relationship is depends on relationship. It is a relationship between use cases indicating that one use case cannot be performed until another use case has been performed **[17]**. Similar to associations, it is symbolized as an arrowhead, but with a text "<<depends on>>" near it.