CS47 - Capstone Project
Las Positas College
Instructor: Carlos Moreno

The Market

Shahaf Dan

May 2020

Contents

1. Abstract

The CS47 (Capstone Project) class at Las Positas College is a self managed and produced project class, in which, along with the assistance of the academic instructor, the students will work over a period of the academic term to produce a final product which was agreed upon with the class instructor.

The Market Manager is the project done for this class by Shahaf Dan for the academic term of the spring of 2020. The project is a web development app including technologies of files output, user input, database programming and managements, front end and back end development with HTML, JavaScript, CSS, PHP, MySQL, and additional plug-ins.

The project is based on the need for a better management system for **The Market**: a monthly event ran by the Student Government at Las Positas College, which in collaboration with the Alameda County food bank provides pabulum, food, and groceries to any attendees from the communities surrounding Las Positas College.

The project includes features such as registration and recorded sign in to all patrons and attendees, management of inventory and records by the specified admin, as well as report generation in a PDF format as well as graphics to help illustrate statistics about the Market.
the project was managed, overlooked, back-upped, stored, and publicly available and accessible through a public and shared GitHub repository.

2. Project Overview

The project's main purpose and objective is to resolve issues faced by the Student Government in running the monthly even known as **the Market**. The Market is a monthly service provided the Las Positas College Student Government, in which, by collaborating with the Alameda County Food Bank, the student organization provides free basic groceries and food to anyone fro communities surrounding the Las Positas campus. The project was constantly publicly shared and stored in the GitHub repository github.com/ShahafDan25/LPCCapstone under the account ShahafDan25

2.1 Purpose

As the Student Government organization at Las Positas has been facing serious problems in managing and tracking event participation, attendance, inventory of the distributed items, as well as registration of attendees, the **Market Manager**, produced as the final product for this course, offers an automated and technological solution in the form of a web application and service. The main features offered in the **Market Manager** include the followings:

- Convenient and easy registration forms for the incoming attendees and patrons.
- A dynamic and flexible report of every record of the market along with graphics and charts to illustrate statistics.
- An option to create a PDF summarized report of any market at any given moment.
- An inventory page to manage unused and undistributed items from previous and current markets.

2.2 Description

The project was developed to assist in the overall and general management of the monthly event offered by the LPCSG [Las Positas College Student Government]. With the development of front-end, back-end, and database structure of the Market Manager, the market could now be overlooked, modified, and changed accordingly to the provided statistics and reports, creating a much more conveniently and effectively managed event to provide better services to the public, reach to larger communities, increase the number of participants, and create a comfortable approach to the organization of the event.

The development of the project included the use of Hyper Test Markup Language [HTML], along with Cascaded Style Sheet and JavaScript for the front-end and interface structure of the project. In addition, libraries such as Morris.js, Bootstrap, and JQuery were used in the development of the user interface. The back-end structure of the application included the use of PHP and PDO databasde connection to a MySQL database running on the localhost, as well as libraries such as FPDF to generate the printable reports.

2.3 Modular Parts

The project was divided into three milestones, or modular parts, each supposedly due at a certain time of the academic term, and by which the developer student had to complete a significant part of his or her project. The modular parts dividing this project included:

Modular Parts		
Milestone 1	Milestone 2	Milestone 3
Registration	**Report**	**Inventory**
index.php	report.php, admin.php	inventory.php
Due February 24	Due April 6	Due May 4

Registration: Provides attendees with a registration (sign in) form both for returning and new members. The page provides access to the admin page via password prompt.

Admin Page: Provides overall look and operation of all markets: creation, deletion, activation, termination, report generation, and inventory management.

report.php: dynamic and live data of every market at any given moment in an organized chart, with graphs to illustrate various statistics. Provides option to create a printable PDF report.

indentory.php Manage and record inventory from previous and current markets. Allows the addition and edit of new and repeating items distributed at the market dynamically at any given moment.

3. Process and Progress

3.1 chronological progress

The progress, as mentioned the in the **Modular Parts** section [2.3], the development progress of the product was divided into three major milestones. However smaller, yet not negligible, additional contributions were made during the time line in regardless of the milestones determined in the project proposal. The project started with the development of the idea, and the recognition for an automated technological solution to improve the efficiency of the monthly service provided by the LPCSG.

After submitting the project proposal and getting the approval by the academic instructor, the first step was to design ahead the project, and create a master plan to ease the development process. Prior to the development process, all files had to be located in a psceific path (**/Library/WebServer/Documents/CapProj**), to allow the dynamic run of all code on the machine's local host (**address 127.0.0.1**). To east the internal navigation on each machine startup, a soft link was created to the local user's desktop. Furthermore, this initial step included the design and structuring of the general interface of the application.

After finalizing the idea and the general look of the product, the development process had begun and included the creation of any files related to the registration page, as well as any files that will serve a general purpose, such as a master CSS file, a master JavaScript file, and a master PHP (back-end) file, whose code will be related and included by all other files.

capstone.css capstone.js capstone.php index.html

The file index.html is the registration page, including a reference to all the general files, in which the CSS, JavaScript, and PHP programs were coded. Ad additional link to a **Bootstrap** CSS library was included. The file did not specify a market date, because it was still generally developed. Using button collapse, the file offers the option to either log in as a returning member, or as a new member. A new member would have to fill out information including a designated identification number, number of people in the household (divided into age categories), and optionally contact information. The information is stored in the local database so any returning patrons to the market could easily sign in by inserting their identification number.

Prior to the deadline of the first milestone, the file index.html became index.php, which provided the filesystem wit the ability to populate information from the database directly into that file, with embedded HTML code. This transformation created the need to a general connection file to the database, which will simply include one function that will return a database connection, and could be conveniently called at any moment from any PHP file: connDB.php, which included the following function alone.

```
function connDB()
{
    $username = "root";
    $password = "MMB3189@A";
    $dsn = 'mysql:dbname=TheMarket;host=127.0.0.1;port=3306;socket=/tmp/mysql.sock';
    //Connection Strine ^ ^
    try
    {
        $conn = new PDO($dsn, $username, $password);
```

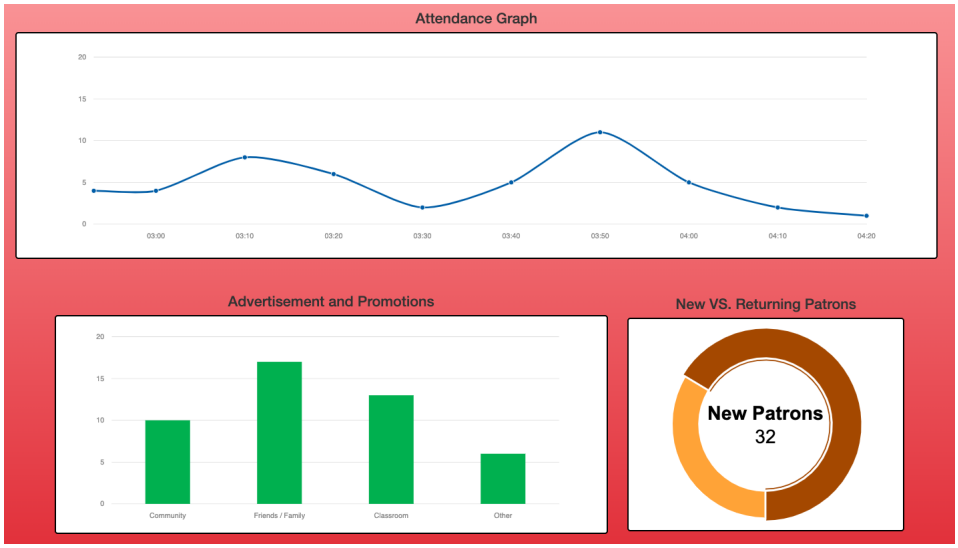
```
    }
    catch (PDOException $e)
    {
        echo 'Connection Failed: ' . $e -> getMessage();
    }
    return $conn;
}
```

At the end of milestone one, the registration page was ready, and included the input and (partial) output of any patron who has ever attended the event.

The development process continue into the second milestone, which included first of all the development of the admin page admin.html, which could be directed and reached by a button click on the index.php (registration) page. The entirety of admin.php would be managed in the back end via the adminFuncs.php file, which included the connection to the database by calling to the connDB() function from the connDB.php file. The admin page allowed the creation of any new market for future or current use, and accordingly added it to the database and the date to the currently displayed and active market. The admin page quickly became the navigation system of the entire software, allowing the (future) movement from inventory, to report, to registration page, of multiple markets simultaneously. At that time of the development process, the admin page had **three main functionalities**: invoke a market, go to the report page, and create a new market based on date.

The main purpose of to achieve during the second milestone was the completion of the report page. As the admin page allowed the navigation to the report page for a specific market, the file report.php would logically need to include the same information as the backend file adminFuncs.php. Therefore, it became logical to make adminFuncs.php the general back-end programming file for all HTML and front end embedded PHP files. Therefore, all code from the capstone.php file was transferred to the adminFuncs.php file.

The report page, which was directed to by choosing a market and the "generate a report" option from the admin page, included information and statistics in nice formatted tables and graphs about the attendees of the selected market. Statistics included the average number of children, adults, and seniors, per household registered, and information such as hourly rate of attendance, returning patrons versus new members, and the promotion method through which the new members have heard about the market, which was all generated in graphs using the **morris.js** graphics tool.



The data for each graph was returned by a. designated function in the admin.funcs file, which created a matrix (double array) of information returned to the **morris.js** script, from which it was directed to a designated HTML division structure to draw the graph. For instance, here is the code used to derive the data needed to compare returning patrons versus new attendees to the market, using embedded SQL query statements.

```
 function getRetVSNew($c, $d) //data for the comparison of returning vs new patrons
{
    $data = "";
    $sql_newPs = "SELECT COUNT(*) FROM Patrons WHERE firstMarket = ".$d.";";
    $s_newPs = $c -> query($sql_newPs);
    $noobies = $s_newPs -> fetchColumn();

    $sql_allPs = "SELECT COUNT(*) FROM MarketLogins WHERE Markets_idByDate = ".$d.";";
    $s_allPs = $c -> query($sql_allPs);
    $allies = $s_allPs -> fetchColumn();

    $data = "{value: ".$noobies.", label: 'New Patrons'},
    {value: ".($allies - $noobies).", label: 'Returning Patrons'}";
    return $data;
}
```

Additionally, the report page provides the admin (the master user) at any given moment, to generate a PDF shortened and summarized reports of all attendees, their basic information, and basic statistics, along with appropriate space for any needed signatures, and the logo of the college and the Student Government displayed at the top of the report, all in a PDF format. The printable option of the report is available thanks to the FPDF library, and the programmed extended class in the moreFPDF.php file.

```
require "otherFiles/fpdf_lib/fpdf.php";
class myFPDFClass extends FPDF //extends the FPDF class with modifications
{
    function Heads($c) {...}
    function tableHead() {...}
    function tableBody($c) {...}
    function signature() {...}
}
```

After the report page, the last part of the project was needed to be programmed: the inventory manager. The inventory is supposed to ease the process of recording all items that were left from each market, and accordingly record it in the database. The purpose is to create a general picture of what groceries and items are used the most, and coordinate with food bank for future markets to order only the groceries that are requested the most, to satisfy an effective supply and demand graph.

The developmet of the inventory end of the software was relatively challenging, considering an 'edit' form for each recorded item needed to be inserted in each form, to allow user-friendly access to editing any records already made to the database. The problem was solved by coding the desired HTML form in the general PHP file and simply echo it by a button click in the foremost form.

```
function populateItemTable($c) //insert items to inventory html in table format
{
    $tableItemData = "";
    $sql = "SELECT DISTINCT Name FROM Items WHERE Markets_idByDate <= (SELECT
```

```
    idByDate FROM Markets WHERE inventory = 1);";
$s_ec = $c -> prepare($sql);
$s_ec -> execute();
$r_ec = $s_ec -> fetch(PDO::FETCH_ASSOC);
if(count($r_ec) == 0)  return '<p> NO ITEMS TO DISPLAY AT THE MOMENT </p>';

$s = $c -> prepare($sql);
$s -> execute();
$collapseCounter = 0;
while($r = $s -> fetch(PDO::FETCH_ASSOC))
{
    //select product's amount from previous markets
    $total = 0;
    $sql_total = "SELECT Amount FROM Items WHERE Name = '".$r['Name']."'
        AND Markets_idByDate < (SELECT idByDate FROM Markets WHERE inventory = 1);";
    $s_total = $c -> prepare($sql_total);
    $s_total -> execute();
    while($r_total = $s_total -> fetch(PDO::FETCH_ASSOC))
    {
        $total += $r_total['Amount'];
    }

    $sql_two = "SELECT Amount FROM Items WHERE Markets_idByDate = (SELECT
        idByDate FROM Markets WHERE inventory = 1) AND Name = '".$r['Name']."';";
    $s_two = $c -> prepare($sql_two);
    $s_two -> execute();
    $r_two = $s_two -> fetch(PDO::FETCH_ASSOC);


    $counter++; //to modify and create unique ID's reflexibly
    $buttonInsert = "<button class = 'btn btn-warning collapsed' id = 'editbtn'
        data-toggle='collapse' data-target='#formToEditItem".strval($counter)."'
        aria-expanded='false'> EDIT </button>";
    //add option to edit an item
    $editForm = '<form action = "adminFuncs.php" method = "post">';
    $editForm .= '<td><input type = "text" class = "inv_input inline" name =
        "editItemName" placeholder = "'.$r['Name'].'"
        value = "'.$r['Name'].'" style = "width: 60% !important;"></td>';
    $editForm .= '<td><input type = "number" class = "inv_input inline" name =
        "editItemAmount" min = "0" placeholder = "'.($r_two['Amount']+$total).'"
        value = "'.($r_two['Amount']+$total).'"></td>';
    $editForm .= '<td><input type = "hidden" class = "inline" name = "message"
    value = "editThatItem">';
    $editForm .= '<button class = "btn btn-success inv_input_btn inline"> SUBMIT
        </button></td>';
    $editForm .= '</form>';
    // ============================================================ //
    $tableItemData .= "<tr>";
    $tableItemData .= "<td>".$r['Name']."</td>";
    if($r_two != 0) {$tableItemData .= "<td><strong>".($r_two['Amount'] + $total).
            "</strong>\t[ ".$r_two['Amount']." + ".$total." * ]</td>";}
```

```
        else {$tableItemData .= "<td><strong>".($total)."</strong>\t*</td>";}
            $tableItemData .= "<td>".$buttonInsert."</td>";
        $tableItemData .= "</tr>";

        $tableItemData .= "<tr class = 'collapse whiteOut' id = 'formToEditItem".strval
            ($counter)."'>".$editForm."</tr>";
    }

    return $tableItemData;
}
```

Along with the development of the inventory, the following files were devloped as well:

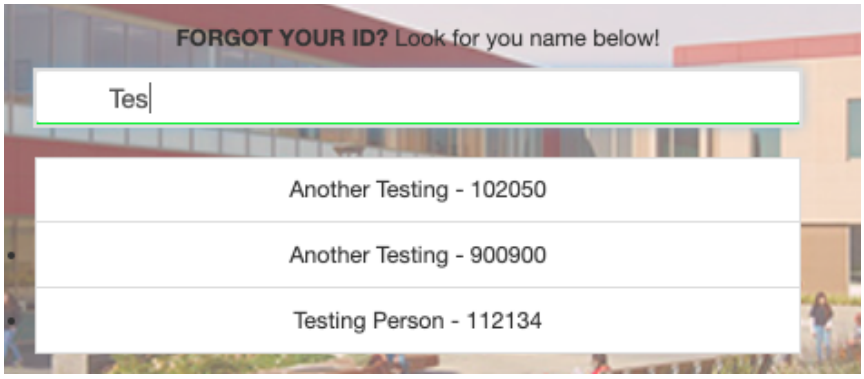
noActiveMarket.html and noCurrentReport.html

each file is the displayed HTML interface the user would see in case either a market is not active and the system navigated to the registration page, or in case the report page was selected from the admin page, but the selected market does not contain any data to display.

Additionally, this part of the software development included the reorganization of the code as a whole both on GitHub, and in the code itself (the restructuring of comments and file system).
The last week prior to the submission of the third and last milestone, minor changes were made to the user interface as a whole, and significant updates were made to registration page, allowing easier access to users who may have forgotten their identification number, by using the **.keyup** function in JavaScript.

```
$(document).ready(function(){
    $("#myInput").on("keyup", function() {
        var value = $(this).val().toLowerCase();
        $("#myList li").filter(function() {
        $(this).toggle($(this).text().toLowerCase().indexOf(value) > -1)
        });
    });
});
```

The snippet code above is an example of dynamic JavaScript, in which as the user types in a specified input bar (HTML input field) a search text, it would search from an array or results and would returned the most relevant searches.



Future additions to the software include the option to sign up as a volunteer, the generation of an email list of patrons, and more.

3.2 Tools and Environments

The development of the market manager software required the integration of many software tools, libraries, languages, techniques, and environments.

FRONT END | INTERFACE

As most web applications, the product of this course was developed in the front end with Hyper Text Markup Language (HTML), which allows the restructuring of basic design element in a flexible design of an HTML page.The front end was designed using cascade Style Sheets (CSS) and a bootstrap library which provides an integrated class of basic CSS styling with the application and use of HTML classes. For the functionality of the site, JavaScript was used along with JQuery code, provided by bootstrap as well.An additional tool used for the design and development of the front end is Morris.js, which allowed the population of HTML DIV (divisions) structures with data sent from the back-end, and designed appropriately by the JavaScript library.

BACK END | FUNCTIONALITY AND COMMUNICATIONS

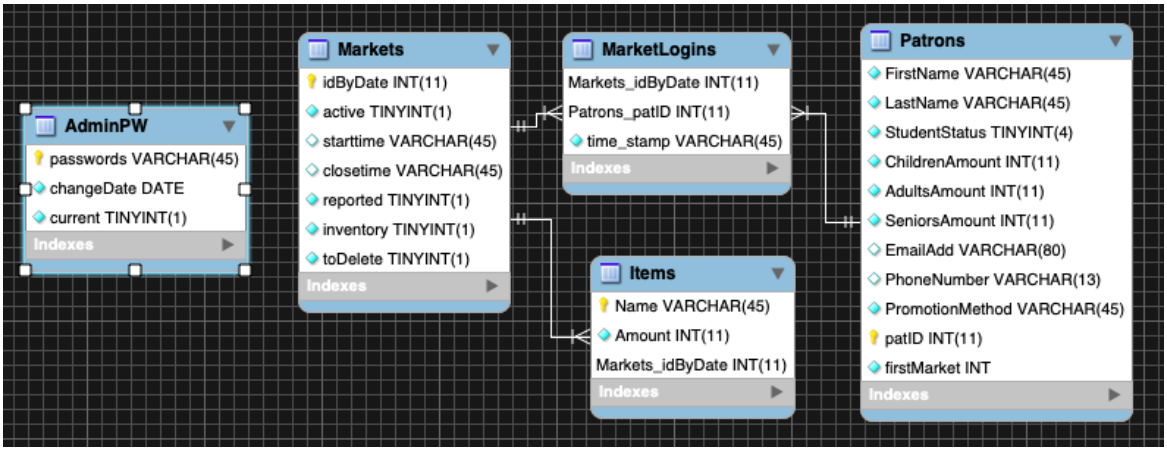
The most convenient development language for back end programming for the web, is often claimed to be PHP [Personal Home Page], which can conveniently conect database data query to HTML structures as output, and from HTML forms as input. Using embedded SQL query lines and the PDO database connection, the main PHP page of the software (the file adminFuncs.php) was able to send and request the desired information from the database.

```
<form class = "inline" id = "someForm" method="post" action="adminFuncs.php">
```

The code snippet is an example of an HTML for dedicated to send information to the PHP page adminFuncs.php, from which manipulation of the sent data will be made and then inserted, recorded or updated to the database.

DATABASES AND SERVERS | DATA INTEGRATION

The database was designed and managed using the MySQL Workbench, which allowed a convenient and friendly design of each entity and its fields, as well as its modification based on the development of the project. As the software constant ran at the localhost (127.0.0.1), the system never experienced more than a single page request at the same time, and therefore there was no need to deal with great traffic or user functionality. If the software will see light in public use in the future, there may need to be made some modification to the code in order to assure correct transactions to manage potential fall-backs and failures of the system at any given moment at run time. Below is a figure describing the designed database.



PROGRAMMING ENVIRONMENTS

The entirty of the programming was done with the texteditor **VisualStudio Code**, which provided with a plug in that allowed the pre-upload display of any HTML file and code. The file system was managed manually via the terminal (command prompt). The database was designed in MySQL workbench, and any test queries were done through the terminal in the MySQL shell (executed by the command **mysqlsh**)

3.3 Resolving Issues, Debugging, Testing

Resolving issues and the debugging of the code were done all mainly by the method of isolation: the problem at which a problem or bug occurs, is found, and isoltaed, often copied to an additional file, in which the code is fixed and debugged. In the case of this project, an additional file called test.php was developed for this exact purpose. Any problem with data collection, query testing, generation of arrays of data manipulation, was solved by copying the code to the test.php file, in which it was modified modular until the problem was resolved.

During the development of the project, most issues were faced in the stages and use of technology with which I had no experience, such as **morris.js** graphs, the **FPDF** library, and **PDO** database connectivity. Resolving this issues had helped me to develop self-teach skills and showed me that the internet has much more resources for young programmers like me, and allowed me to practice my skills of searching for software and program bugs solutions online.

As part of the third milestone, I had also tested the project by creating multiple markets and inserting great amount of data to the database, and checking the functionality of the software in every single file. In other words, I had simulated the software as if it were to be used by the student government body at the upcoming monthly market service.

To enter the great amount of data to the database, I simply created an **AppleScript**, which allows the automation of keyboard typing and mouse movement. By creating large arrays with random first and last names found online, the **AppleScript** populates each active market with random samples of people logging in over a random time range (limited to 200 minutes).

3.4 Outcomes and Future Use

Currently, the website runs properly, and without any interruptions or errors at the localhost. It functions properly, and after broad testing can successfully simulated and illustrate the functionality and management of the monthly service ran by the Student Government at Las Posiats College. In the future, if approved by the student government advisors and executive board, I would hope to upload the site to the **www** via a personal PHP server.If and when the project will be live to help the organization manage the monthly service event, modification will be needed to be made. Some modifications include:

- option to generate an email list of all patrons who attended the market and provided an email address, as well as send emails from within the app to attendees from previous markets and reminders to volunteers.
- A volunteer management page, which allows the scheduling of each volunteer to a specific time slot, and the accessibility of other users besides the admins to the site, allowing them to sign up.
- Integration of the front end based on media size: allowing to open the software on any device of any size, at any given moment.

4. Summary

All in all, after finishing the project and creating a software for good use, I confidently say that this class is surprisingly the class in which I had learned the most, or at least it feels this way. The knowledge I had collected throughout the semester allowed me to create honors projects for other classes, meet the web development world from up-close, and start generating ideas for potential projects I may conduct in the future.

As a matter of fact, I already have an idea I believe could be relatively 'big', and I will work towards programming it over the summer. I now finally feel confident enough to go to hackathons, and believe in my ability to code a software solution within 24 - 48 hours (of intensely focused coding) only, and potentially win hackathons and other programming competitions.

A piece of advise I would leave to future students, is to maintain and constantly keep a log book of any changed made to the project. To my, GitHub commit comments were not enough, so as I am typing this report in the LaTeX text editor, I am having a difficult time remembering everything that I had done. Additioally, maintaining a porject journal is always good, in case anything goes wrong with the code, in which case the log book provides quick access to any possible changes that may have caused this problem.

I am greatly thankful for the help, assistance, and guidance from the academic instructor, Carlos Moreno, who had helped me incredibly in the understanding of the web development technology, and helped in me in many situations in which I was lost, stuck, or simply clueless on what to do. I am now confident that I can also search and self teach myself anything to get out of situations where I fee lost in my own code.

Although I had previously held basic skills of software developments, specifically in front end technology such as the one I used (HTML, JavaScript, CSS, etc), I have learned much more the imagined and am grateful for taking this course. All in all, I highly recommend any young software developers and computer science student at Las Posiats College to take this class is they can afford 3 additional unites, as it will enrich their knowledge and provide them with skills applicable to their field of interest (related to computer science) and will allow them to see there so much more to computer science than pointers, classes, data structures, and binary digits.