BAIT3173 Integrative Programming

ASSIGNMENT 202301

Student Name	:	Lee Ting Le
Student ID	:	22PMR06024
Programme	:	RSD
Tutorial Group	:	G2
System Title	:	Japanese Online Ticketing System
Modules	:	Event, Ticket, Category

Declaration

- I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University College's plagiarism policy.
- I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

[illegible]

Student's signature

12 May 2023

Date

Table of Contents

1. Introduction to the System

The system is a Japan Online Event Ticketing System that will be used by the organization Kokoro to allow customers to electronically book tickets for ongoing and future events held by Kokoro. Kokoro is a commercial and for-profit organization where it sells Japanese food products and is looking into expanding its line of products by creating Japanese related events for people that are interested in Japanese culture hence why they require this system. The organization was founded in 2020 by Miss Lim En Xi.

The system is developed from multiple programming such as PHP and XML, uses OOP concepts and SQL as the database for the system. In the system, the user interfaces included for customer interaction are such as home page, a page for lists of events from Kokoro and the details of those events and so on. And this system's main purpose is for users to view the ongoing and/or upcoming events made by Kokoro and purchase tickets for those events if they are interested in it. The system has different capabilities such as updating the information for the events quite frequently where users are able to see the amount of tickets remaining for them to purchase a ticket for the events they are interested in.

2. Modules

2.1 Module Description

I am responsible for three main modules in the online ticketing system: **Event**, **Ticket**, and **Category**. The Event module is primarily focused on managing and presenting events to users. It stores event details such as the name, description, start and end dates, category, ticket availability, event status, and price in the database. This module supports CRUD operations, allowing creation of new events, retrieval of event details for presentation, updating of existing event information, and deletion of events when necessary. Additionally, it provides a recovery feature that allows staff to restore accidentally deleted events.

The Event module also offers filtering and sorting capabilities, enabling staff members to easily search for events. They can filter events based on criteria like the event name, status, price, and other relevant attributes. Sorting options allow them to organize events by name in various orders. The module ensures that ticket availability is continuously updated, reflecting changes such as tickets selling out or events no longer available. This guarantees that users always receive up-to-date information.

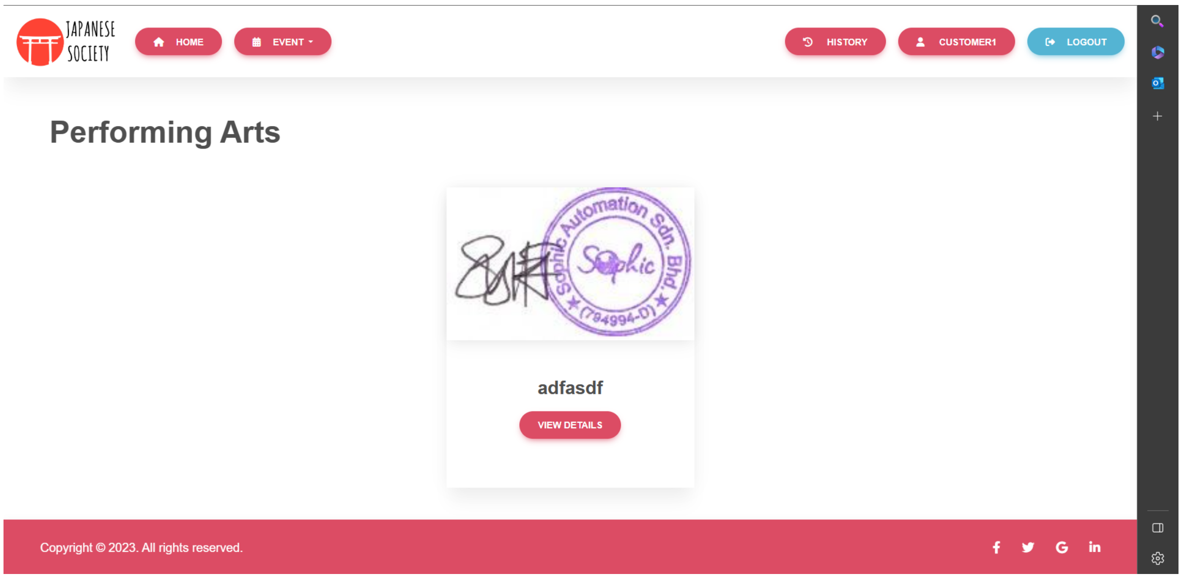
Moving on to the Ticket module, it stores ticket details such as the ticket ID, ticket code, associated event, status, and ticket holder in the database. It is closely linked to the Event module, as it will be initiated only when events are created. The Ticket module generates unique ticket codes for each ticket based on the number of tickets created for an event. It also records the name of the ticket holder for easy identification. Additionally, the module automatically updates the ticket status to 'inactive' when the corresponding event is no longer available. It also provides sorting and filtering functionalities.

Lastly, the Category module handles the categorization of events into different types. It supports common functionalities like CRUD operations, as well as sorting and filtering. The purpose of this module is to allow users to easily differentiate between various event categories.

In summary, my role in the system involves overseeing the core functionalities of creating and managing events, enabling users to make purchases efficiently.

2.2 Module Screenshot

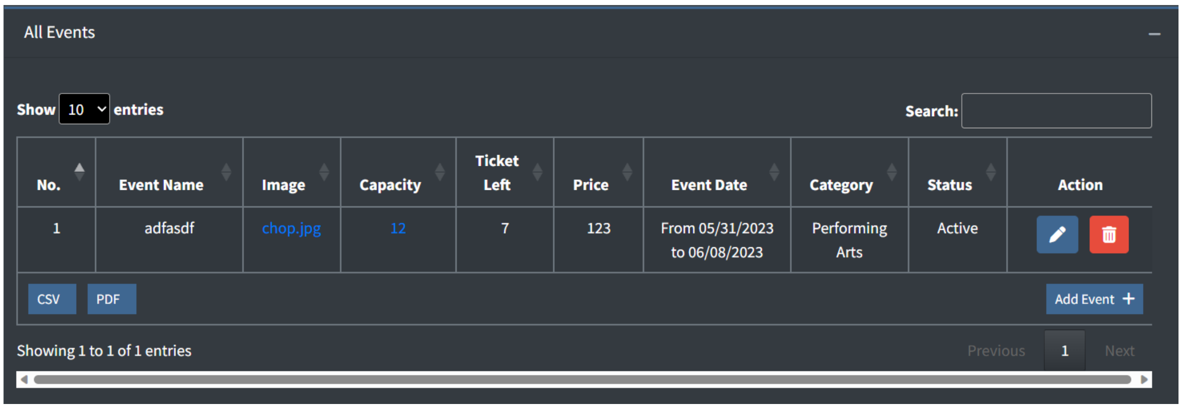
Event



Performing Arts

adfasdf

Copyright © 2023. All rights reserved.



All Events

Show 10 entries

Search:

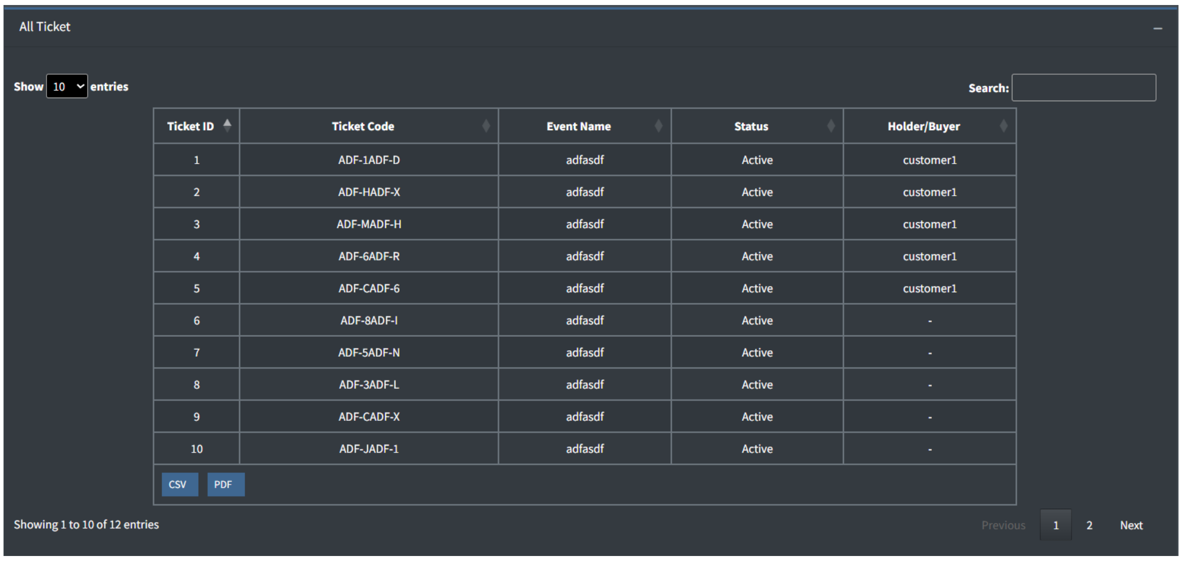
No.	Event Name	Image	Capacity	Ticket Left	Price	Event Date	Category	Status	Action
1	adfasdf	chop.jpg	12	7	123	From 05/31/2023 to 06/08/2023	Performing Arts	Active	

Showing 1 to 1 of 1 entries

- **The first screenshot is showing the list of events that are display to the client**
- **Client can choose to purchase the event**

- **The second screenshot is showing all the event data in table format**
- **User choose to add, edit, delete or recover the event data**
- **User can also download the event data in Excel or PDF format**
- **User can search for the particular event information too**
- **User can also filtering how many event records will be show**

Ticket



All Ticket

Show 10 entries

Search:

Ticket ID	Ticket Code	Event Name	Status	Holder/Buyer
1	ADF-1ADF-D	adfasdf	Active	customer1
2	ADF-HADF-X	adfasdf	Active	customer1
3	ADF-MADF-H	adfasdf	Active	customer1
4	ADF-6ADF-R	adfasdf	Active	customer1
5	ADF-CADF-6	adfasdf	Active	customer1
6	ADF-8ADF-I	adfasdf	Active	-
7	ADF-5ADF-N	adfasdf	Active	-
8	ADF-3ADF-L	adfasdf	Active	-
9	ADF-CADF-X	adfasdf	Active	-
10	ADF-JADF-1	adfasdf	Active	-

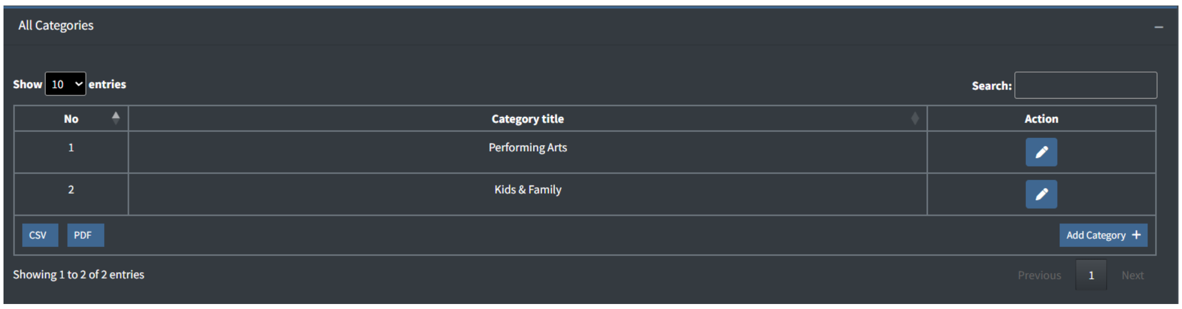
CSV PDF

Showing 1 to 10 of 12 entries

Previous 1 2 Next

- This screenshot is showing all the ticket data in table format
- User choose to add, edit, delete or recover the ticket data
- User can also download the ticket data in Excel or PDF format
- User can search for the particular ticket information too
- User can also filtering how many ticket records will be show

Category



All Categories

Show 10 entries

Search:

No	Category title	Action
1	Performing Arts	✎
2	Kids & Family	✎

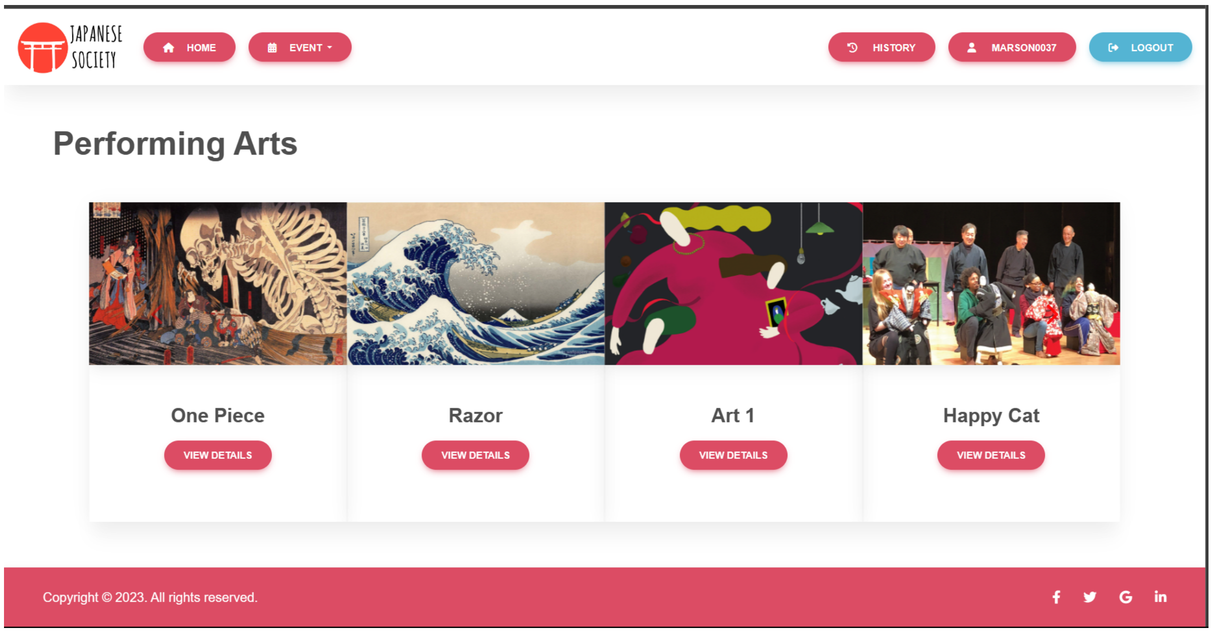
CSV PDF Add Category +

Showing 1 to 2 of 2 entries

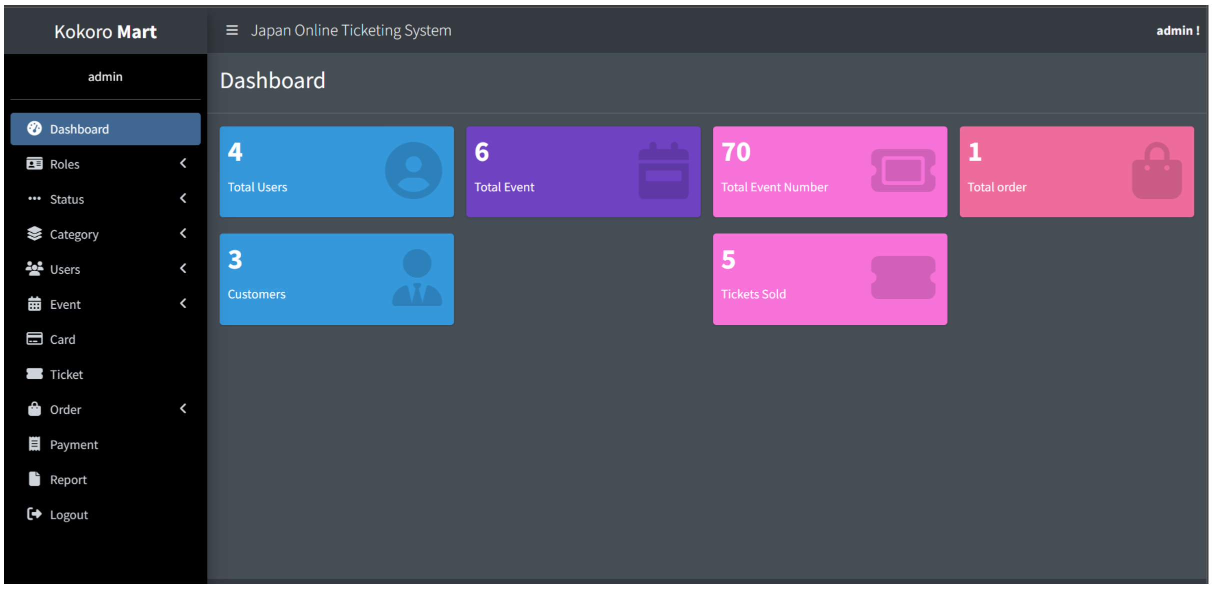
Previous 1 Next

- This screenshot is showing all the category data in table format
- User choose to add, edit, delete or recover the category data
- User can also download the category data in Excel or PDF format
- User can search for the particular category information too
- User can also filtering how many category records will be show

2.3 Extra Screenshot



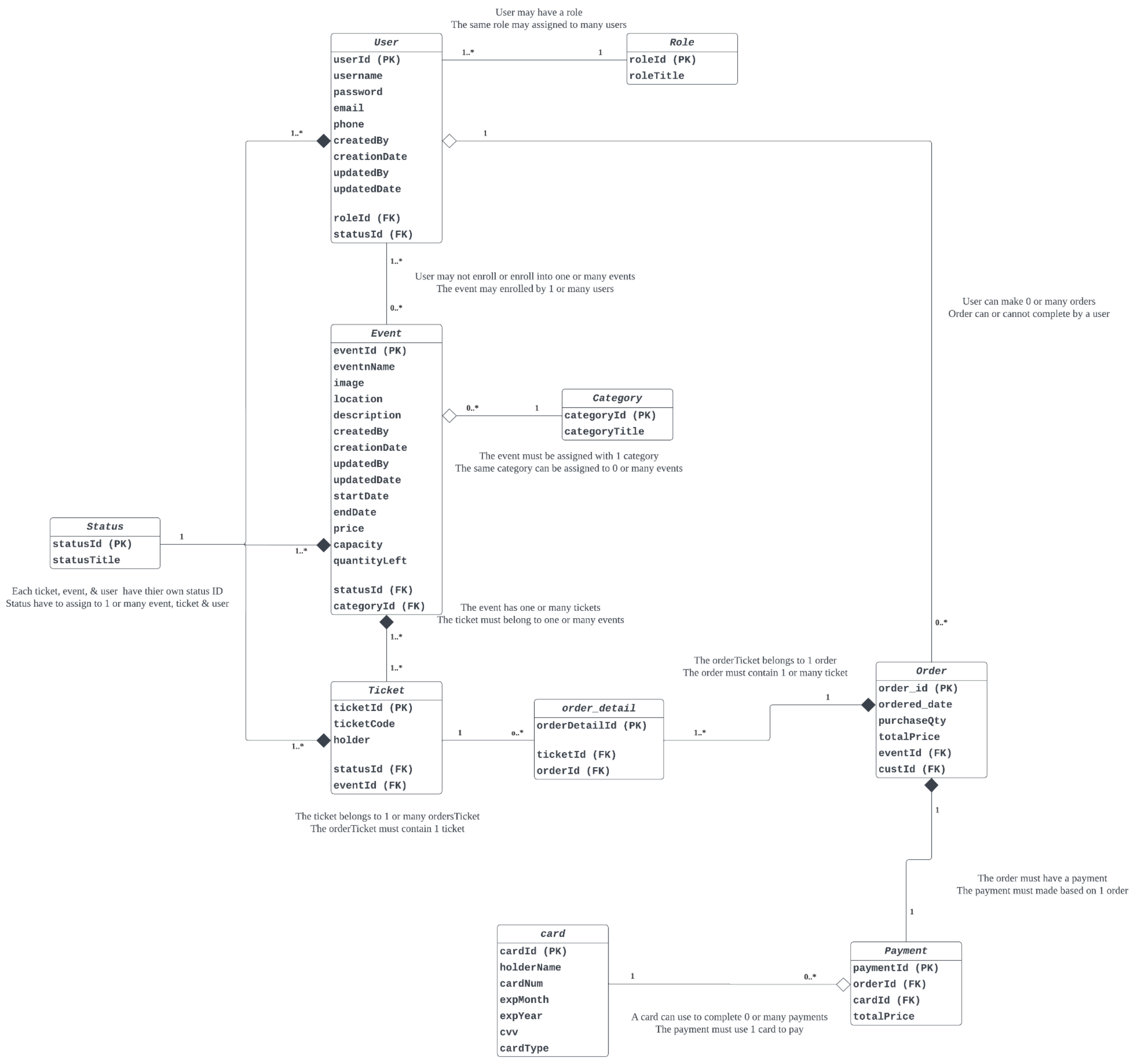
This is the event list in card format which allows users to choose.



Above screenshot will show the overall details of the website.

3. Entity Classes

3.1 Entity Class Diagram


User may have a role
The same role may assigned to many users
User
userId (PK)
username
password
email
phone
createdBy
creationDate
updatedBy
updatedDate
roleId (FK)
statusId (FK)
Role
roleId (PK)
roleTitle
1..*
1
1..*
1
User may not enroll or enroll into one or many events
The event may enrolled by 1 or many users
0..*
User can make 0 or many orders
Order can or cannot complete by a user
Event
eventId (PK)
eventnName
image
location
description
createdBy
creationDate
updatedBy
updatedDate
startDate
endDate
price
capacity
quantityLeft
statusId (FK)
categoryId (FK)
0..*
Category
categoryId (PK)
categoryTitle
The event must be assigned with 1 category
The same category can be assigned to 0 or many events
Status
statusId (PK)
statusTitle
Each ticket, event, & user have thier own status ID
Status have to assign to 1 or many event, ticket & user
The event has one or many tickets
The ticket must belong to one or many events
The orderTicket belongs to 1 order
The order must contain 1 or many ticket
Ticket
ticketId (PK)
ticketCode
holder
statusId (FK)
eventId (FK)
order_detail
orderDetailId (PK)
ticketId (FK)
orderId (FK)
Order
order_id (PK)
ordered_date
purchaseQty
totalPrice
eventId (FK)
custId (FK)
The ticket belongs to 1 or many ordersTicket
The orderTicket must contain 1 ticket
The order must have a payment
The payment must made based on 1 order
card
cardId (PK)
holderName
cardNum
expMonth
expYear
cvv
cardType
Payment
paymentId (PK)
orderId (FK)
cardId (FK)
totalPrice
A card can use to complete 0 or many payments
The payment must use 1 card to pay

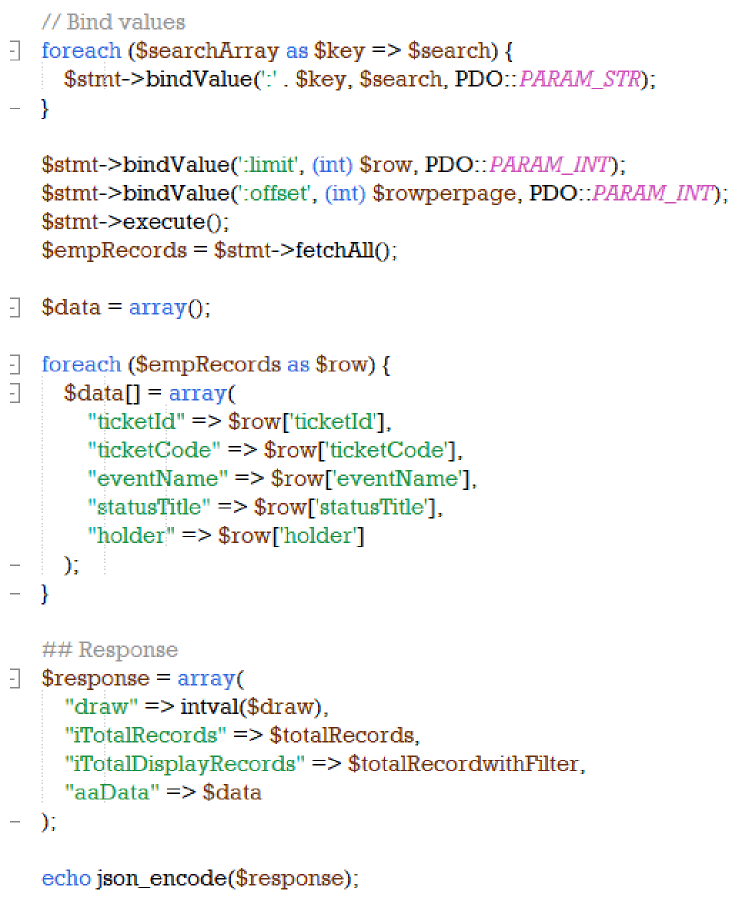
3.2 Entity Classes

3.2.1 Ticket Module

ticket_views.php

- **Retrieve and process data in data table format, then return it to the html via ajax response.**
- **Apply sorting, filtering, searching and pagination on the data too**

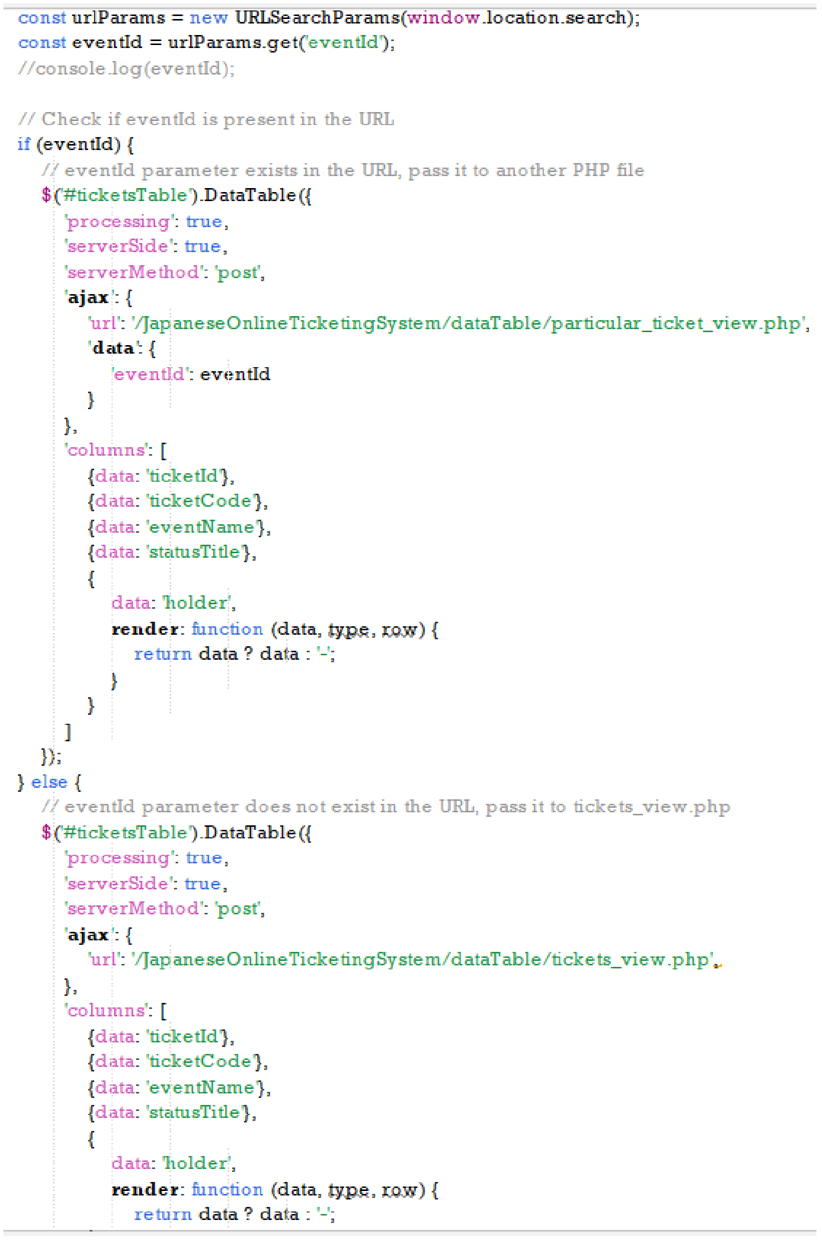
```
## Search
$searchQuery = " ";
if ($searchValue != '') {
  $searchQuery = " AND (t.ticketCode LIKE :ticketCode)
           OR (t.holder LIKE :holder)
           OR (e.eventName LIKE :eventName)
           OR (s.statusTitle LIKE :statusTitle) ";
  $searchArray = array(
    'ticketCode' => "%$searchValue%",
    'holder' => "%$searchValue%",
    'eventName' => "%$searchValue%",
    'statusTitle' => "%$searchValue%"
  );
}

## Total number of records without filtering
$stmt = $conn->prepare("SELECT COUNT(*) AS allcount FROM ticket ");
$stmt->execute();
$records = $stmt->fetch();
$totalRecords = $records['allcount'];

## Total number of records with filtering
$stmt = $conn->prepare("SELECT COUNT(*) AS allcount
             FROM ticket AS t
            JOIN status AS s ON t.statusId=s.statusId
            JOIN event AS e ON t.eventId = e.eventId
            WHERE 1 " . $searchQuery);
$stmt->execute($searchArray);
$records = $stmt->fetch();
$totalRecordwithFilter = $records['allcount'];

## Fetch records
$stmt = $conn->prepare("SELECT t.*, s.statusTitle, e.eventName
    FROM ticket AS t
    JOIN status AS s ON t.statusId=s.statusId
    JOIN event AS e ON t.eventId=e.eventId
    WHERE 1 " . $searchQuery . "
    ORDER BY " . $columnName . " " . $columnSortOrder . "
    LIMIT :limit,:offset");
```



```php
// Bind values
foreach ($searchArray as $key => $search) {
    $stmt->bindValue(':' . $key, $search, PDO::PARAM_STR);
}

$stmt->bindValue(':limit', (int) $row, PDO::PARAM_INT);
$stmt->bindValue(':offset', (int) $rowperpage, PDO::PARAM_INT);
$stmt->execute();
$empRecords = $stmt->fetchAll();

$data = array();

foreach ($empRecords as $row) {
    $data[] = array(
        "ticketId" => $row['ticketId'],
        "ticketCode" => $row['ticketCode'],
        "eventName" => $row['eventName'],
        "statusTitle" => $row['statusTitle'],
        "holder" => $row['holder']
    );
}

## Response
$response = array(
    "draw" => intval($draw),
    "iTotalRecords" => $totalRecords,
    "iTotalDisplayRecords" => $totalRecordwithFilter,
    "aaData" => $data
);

echo json_encode($response);
```

view_tickets_jquery.js

- **Retrieve the processed data from the ticket_views file**
- **Rendered the data in the html table**
- **This jquery page also send an ajax request for processing csv and pdf file using XML (but not gonna show here, and other jquery did the same too)**



```
const urlParams = new URLSearchParams(window.location.search);
const eventId = urlParams.get('eventId');
//console.log(eventId);

// Check if eventId is present in the URL
if (eventId) {
    // eventId parameter exists in the URL, pass it to another PHP file
    $('#ticketsTable').DataTable({
        'processing': true,
        'serverSide': true,
        'serverMethod': 'post',
        'ajax': {
            'url': '/JapaneseOnlineTicketingSystem/dataTable/particular_ticket_view.php',
            'data': {
                'eventId': eventId
            }
        },
        'columns': [
            {data: 'ticketId'},
            {data: 'ticketCode'},
            {data: 'eventName'},
            {data: 'statusTitle'},
            {
                data: 'holder',
                render: function (data, type, row) {
                    return data ? data : '-';
                }
            }
        ]
    });
} else {
    // eventId parameter does not exist in the URL, pass it to tickets_view.php
    $('#ticketsTable').DataTable({
        'processing': true,
        'serverSide': true,
        'serverMethod': 'post',
        'ajax': {
            'url': '/JapaneseOnlineTicketingSystem/dataTable/tickets_view.php',
        },
        'columns': [
            {data: 'ticketId'},
            {data: 'ticketCode'},
            {data: 'eventName'},
            {data: 'statusTitle'},
            {
                data: 'holder',
                render: function (data, type, row) {
                    return data ? data : '-';
```

view_all_ticket.php

- **Contain HTML table that use to display the processed data table**

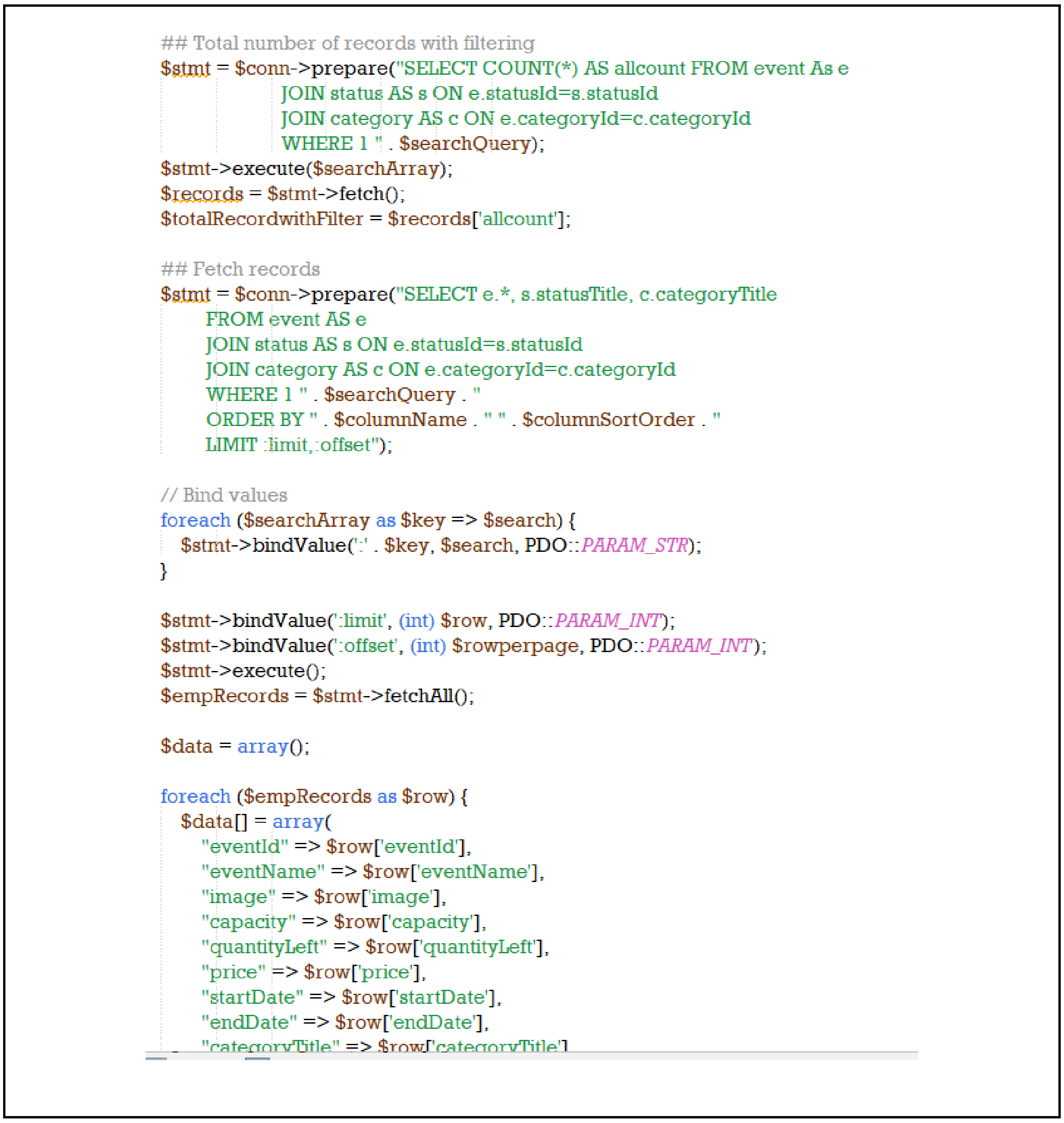
```
  <table id="ticketsTable"
     class="table table-striped dataTable table-bordered dtr-inline text-center"
     role="grid" aria-describedby="all_tickets_info">
    <colgroup>
      <col width="10%">
      <col width="30%">
      <col width="20%">
      <col width="20%">
      <col width="20%">
    </colgroup>
    <thead>
      <tr>
        <th>Ticket ID</th>
        <th>Ticket Code</th>
        <th>Event Name</th>
        <th>Status</th>
        <th>Holder/Buyer</th>
      </tr>
    </thead>
    <tfoot>
      <tr>
        <td colspan="5" class="text-left">
          <a  id="csv" class="btn btn-primary btn-sm btn-flat mb-1">
            <span class="mr-2">CSV</span>
          </a>
          <a id="pdf" class="btn btn-primary btn-sm btn-flat mb-1 ml-2">
            <span class="mr-2">PDF</span>
          </a>
        </td>
      </tr>
    </tfoot>
  </table>
</div>
```

3.2.2 Event Module

events_view.php

- **Same with previous section, this file will retrieve event data from the database and process the data into Datatable format**
- **Apply the features such as filtering, searching, pagination and sorting on the event data**

```
## Search
$searchQuery = " ";
if ($searchValue != '') {
  $searchQuery = " AND (e.eventId LIKE :eventId)
        OR (e.eventName LIKE :eventName)
        OR (e.image LIKE :image)
        OR (e.capacity LIKE :capacity)
        OR (e.quantityLeft LIKE :quantityLeft)
        OR (e.startDate LIKE :startDate)
        OR (e.endDate LIKE :endDate)
        OR (e.price LIKE :price)
        OR (s.statusTitle LIKE :statusTitle)
        OR (c.categoryTitle LIKE :categoryTitle) ";
  $searchArray = array(
    'eventId' => "%$searchValue%",
    'eventName' => "%$searchValue%",
    'image' => "%$searchValue%",
    'capacity' => "%$searchValue%",
    'quantityLeft' => "%$searchValue%",
    'startDate' => "%$searchValue%",
    'endDate' => "%$searchValue%",
    'price' => "%$searchValue%",
    'statusTitle' => "%$searchValue%",
    'categoryTitle' => "%$searchValue%",
  );
}

## Total number of records without filtering
$stmt = $conn->prepare("SELECT COUNT(*) AS allcount FROM event As e");
$stmt->execute();
$records = $stmt->fetch();
$totalRecords = $records['allcount'];

## Total number of records with filtering
$stmt = $conn->prepare("SELECT COUNT(*) AS allcount FROM event As e
            JOIN status AS s ON e.statusId=s.statusId
            JOIN category AS c ON e.categoryId=c.categoryId
            WHERE 1 " . $searchQuery);
$stmt->execute($searchArray);
$records = $stmt->fetch();
$totalRecordwithFilter = $records['allcount'];
```

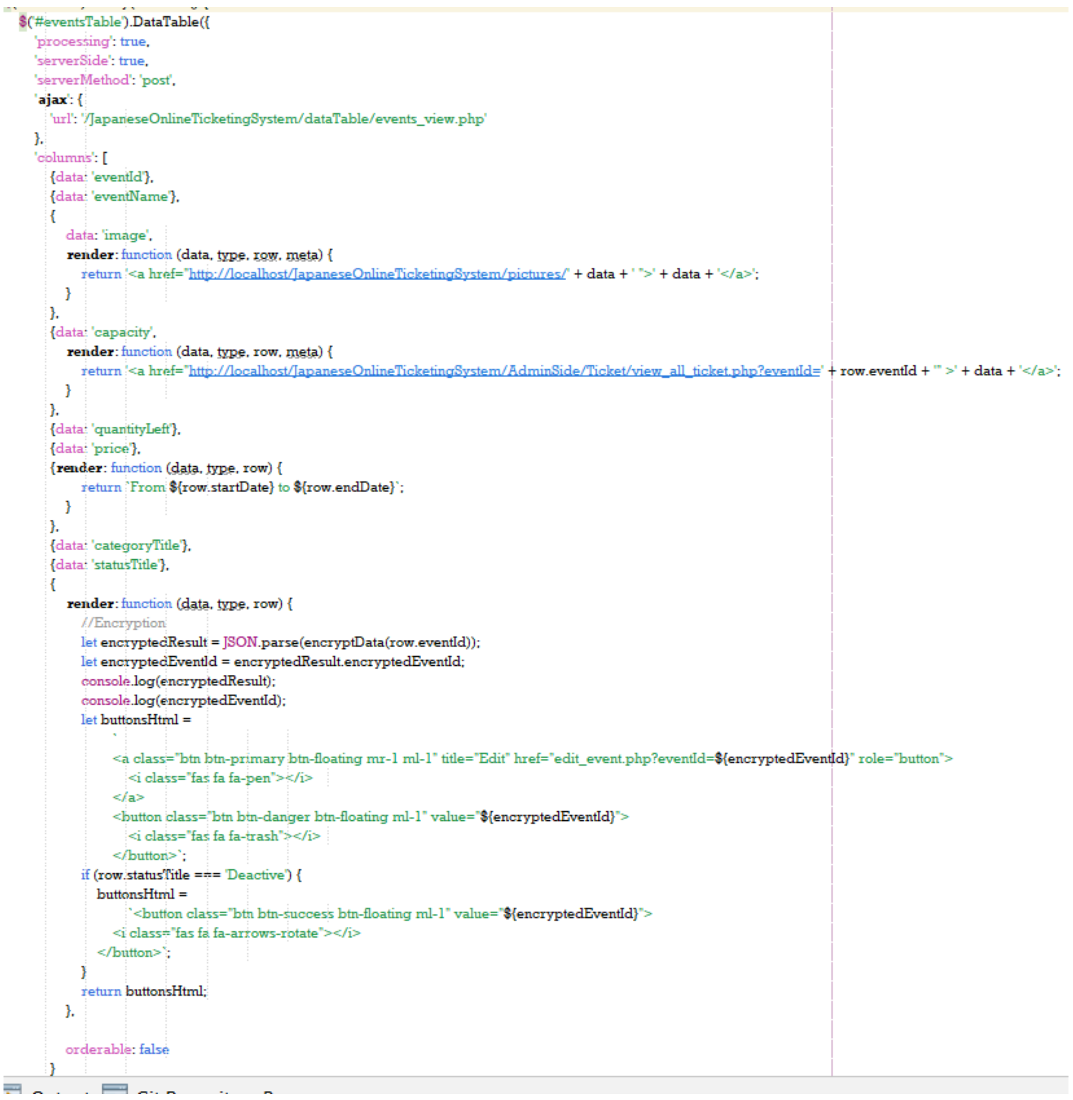


```
## Total number of records with filtering
$stmt = $conn->prepare("SELECT COUNT(*) AS allcount FROM event As e
                    JOIN status AS s ON e.statusId=s.statusId
                    JOIN category AS c ON e.categoryId=c.categoryId
                    WHERE 1 " . $searchQuery);
$stmt->execute($searchArray);
$records = $stmt->fetch();
$totalRecordwithFilter = $records['allcount'];

## Fetch records
$stmt = $conn->prepare("SELECT e.*, s.statusTitle, c.categoryTitle
       FROM event AS e
       JOIN status AS s ON e.statusId=s.statusId
       JOIN category AS c ON e.categoryId=c.categoryId
       WHERE 1 " . $searchQuery . "
       ORDER BY " . $columnName . " " . $columnSortOrder . "
       LIMIT :limit,:offset");

// Bind values
foreach ($searchArray as $key => $search) {
   $stmt->bindValue(':' . $key, $search, PDO::PARAM_STR);
}

$stmt->bindValue(':limit', (int) $row, PDO::PARAM_INT);
$stmt->bindValue(':offset', (int) $rowperpage, PDO::PARAM_INT);
$stmt->execute();
$empRecords = $stmt->fetchAll();

$data = array();

foreach ($empRecords as $row) {
   $data[] = array(
      "eventId" => $row['eventId'],
      "eventName" => $row['eventName'],
      "image" => $row['image'],
      "capacity" => $row['capacity'],
      "quantityLeft" => $row['quantityLeft'],
      "price" => $row['price'],
      "startDate" => $row['startDate'],
      "endDate" => $row['endDate'],
      "categoryTitle" => $row['categoryTitle']
```

```
foreach ($empRecords as $row) {
   $data[] = array(
      "eventId" => $row['eventId'],
      "eventName" => $row['eventName'],
      "image" => $row['image'],
      "capacity" => $row['capacity'],
      "quantityLeft" => $row['quantityLeft'],
      "price" => $row['price'],
      "startDate" => $row['startDate'],
      "endDate" => $row['endDate'],
      "categoryTitle" => $row['categoryTitle'],
      "statusTitle" => $row['statusTitle']
   );
}

## Response
$response = array(
   "draw" => intval($draw),
   "iTotalRecords" => $totalRecords,
   "iTotalDisplayRecords" => $totalRecordwithFilter,
   "aaData" => $data
);

echo json_encode($response);
```

view_event_jquery.js

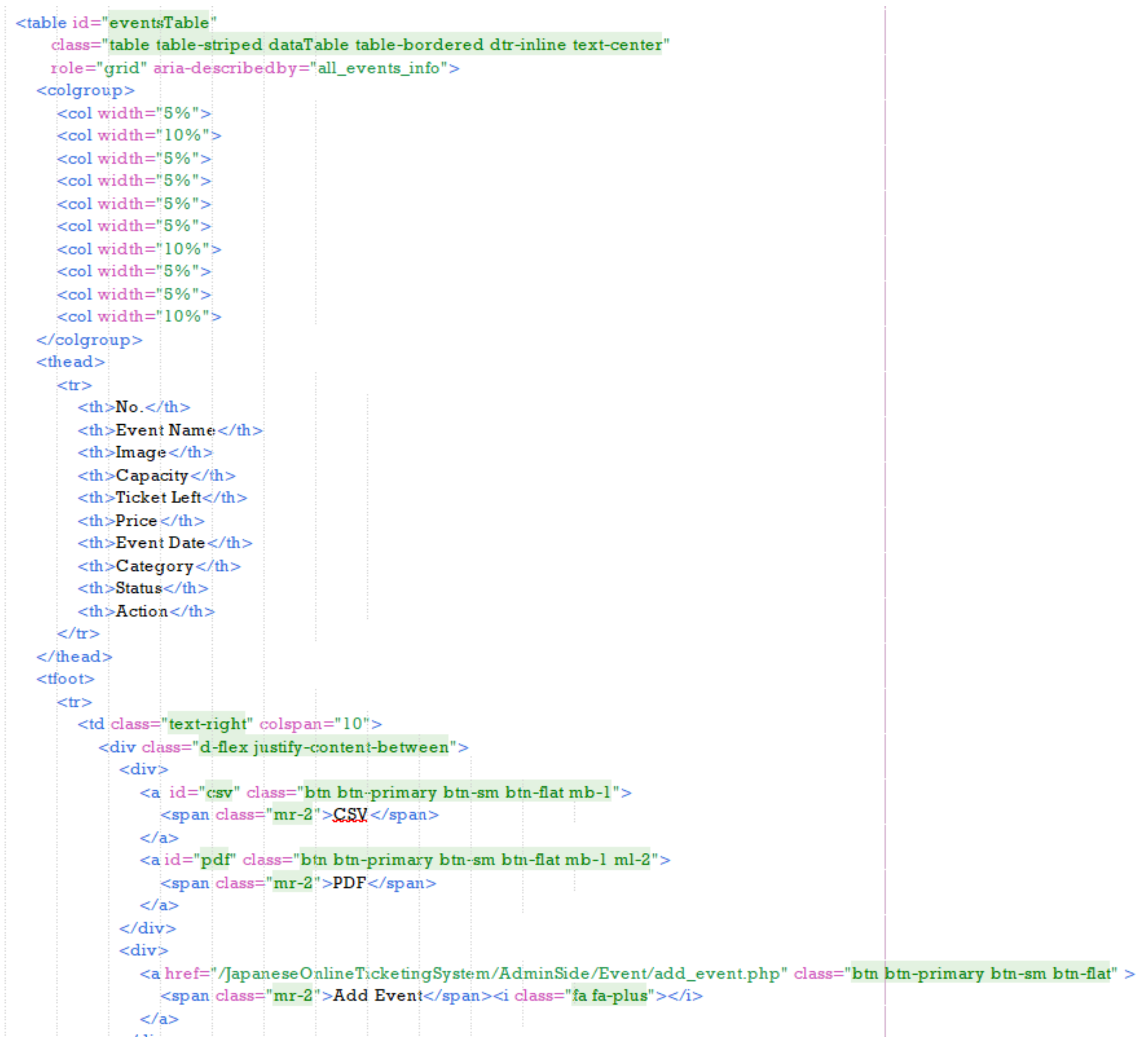
- **Retrieve the processed event Datatable and render it at the html table**



```
$('#eventsTable').DataTable({
    'processing': true,
    'serverSide': true,
    'serverMethod': 'post',
    'ajax': {
        'url': '/JapaneseOnlineTicketingSystem/dataTable/events_view.php'
    },
    'columns': [
        {data: 'eventId'},
        {data: 'eventName'},
        {
            data: 'image',
            render: function (data, type, row, meta) {
                return '<a href="http://localhost/JapaneseOnlineTicketingSystem/pictures/' + data + ' ">' + data + '</a>';
            }
        },
        {data: 'capacity',
            render: function (data, type, row, meta) {
                return '<a href="http://localhost/JapaneseOnlineTicketingSystem/AdminSide/Ticket/view_all_ticket.php?eventId=' + row.eventId + '" >' + data + '</a>';
            }
        },
        {data: 'quantityLeft'},
        {data: 'price'},
        {render: function (data, type, row) {
                return `From ${row.startDate} to ${row.endDate}`;
            }
        },
        {data: 'categoryTitle'},
        {data: 'statusTitle'},
        {
            render: function (data, type, row) {
                //Encryption
                let encryptedResult = JSON.parse(encryptData(row.eventId));
                let encryptedEventId = encryptedResult.encryptedEventId;
                console.log(encryptedResult);
                console.log(encryptedEventId);
                let buttonsHtml =
                        `
                        <a class="btn btn-primary btn-floating mr-1 ml-1" title="Edit" href="edit_event.php?eventId=${encryptedEventId}" role="button">
                            <i class="fas fa fa-pen"></i>
                        </a>
                        <button class="btn btn-danger btn-floating ml-1" value="${encryptedEventId}">
                            <i class="fas fa fa-trash"></i>
                        </button>`;
                if (row.statusTitle === 'Deactive') {
                    buttonsHtml =
                            `<button class="btn btn-success btn-floating ml-1" value="${encryptedEventId}">
                        <i class="fas fa fa-arrows-rotate"></i>
                    </button>`;
                }
                return buttonsHtml;
            },

            orderable: false
        }
```

view_all_events.php

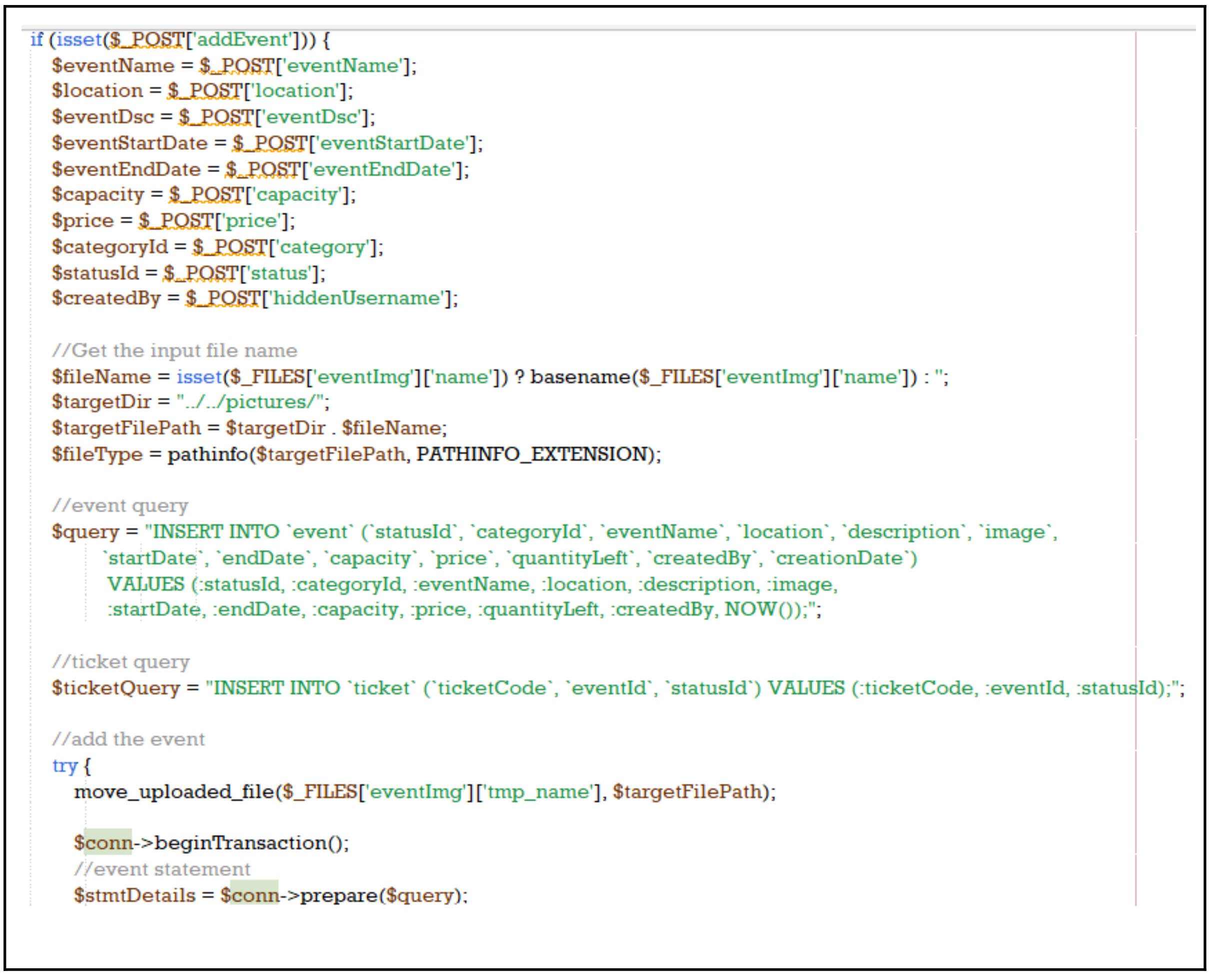
- **Display the processed event Datatable to the user**



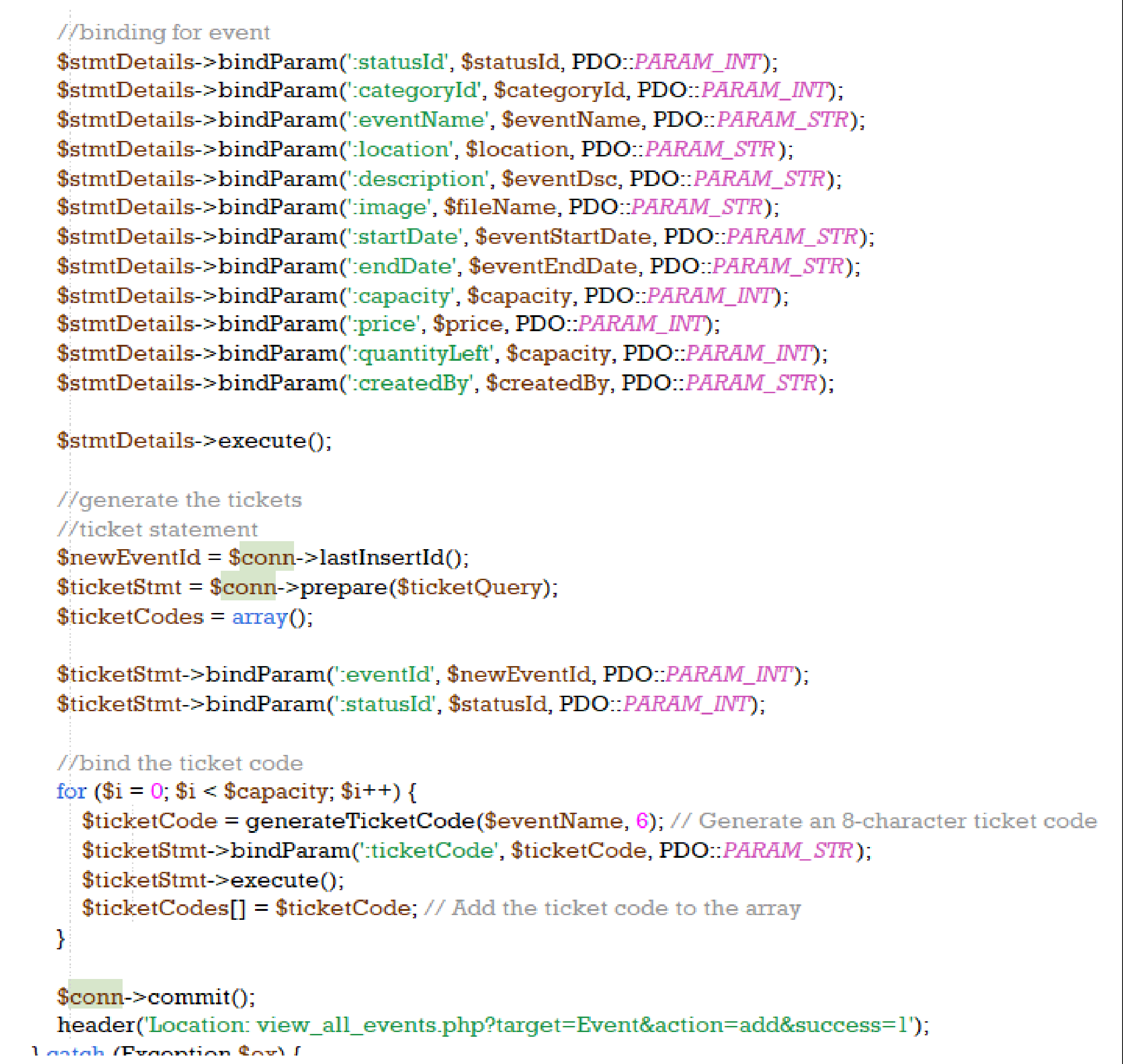
```
<table id="eventsTable"
     class="table table-striped dataTable table-bordered dtr-inline text-center"
     role="grid" aria-describedby="all_events_info">
  <colgroup>
    <col width="5%">
    <col width="10%">
    <col width="5%">
    <col width="5%">
    <col width="5%">
    <col width="5%">
    <col width="10%">
    <col width="5%">
    <col width="5%">
    <col width="10%">
  </colgroup>
  <thead>
    <tr>
      <th>No.</th>
      <th>Event Name</th>
      <th>Image</th>
      <th>Capacity</th>
      <th>Ticket Left</th>
      <th>Price</th>
      <th>Event Date</th>
      <th>Category</th>
      <th>Status</th>
      <th>Action</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td class="text-right" colspan="10">
        <div class="d-flex justify-content-between">
          <div>
            <a  id="csv" class="btn btn-primary btn-sm btn-flat mb-1">
              <span class="mr-2">CSV</span>
            </a>
            <a id="pdf" class="btn btn-primary btn-sm btn-flat mb-1 ml-2">
              <span class="mr-2">PDF</span>
            </a>
          </div>
          <div>
            <a href="/JapaneseOnlineTicketingSystem/AdminSide/Event/add_event.php" class="btn btn-primary btn-sm btn-flat" >
              <span class="mr-2">Add Event</span><i class="fa fa-plus"></i>
            </a>
```

add_event.php

- Allow user to add new event to the database



```php
if (isset($_POST['addEvent'])) {
  $eventName = $_POST['eventName'];
  $location = $_POST['location'];
  $eventDsc = $_POST['eventDsc'];
  $eventStartDate = $_POST['eventStartDate'];
  $eventEndDate = $_POST['eventEndDate'];
  $capacity = $_POST['capacity'];
  $price = $_POST['price'];
  $categoryId = $_POST['category'];
  $statusId = $_POST['status'];
  $createdBy = $_POST['hiddenUsername'];

  //Get the input file name
  $fileName = isset($_FILES['eventImg']['name']) ? basename($_FILES['eventImg']['name']) : '';
  $targetDir = "../../pictures/";
  $targetFilePath = $targetDir . $fileName;
  $fileType = pathinfo($targetFilePath, PATHINFO_EXTENSION);

  //event query
  $query = "INSERT INTO `event` (`statusId`, `categoryId`, `eventName`, `location`, `description`, `image`,
       `startDate`, `endDate`, `capacity`, `price`, `quantityLeft`, `createdBy`, `creationDate`)
       VALUES (:statusId, :categoryId, :eventName, :location, :description, :image,
       :startDate, :endDate, :capacity, :price, :quantityLeft, :createdBy, NOW());";

  //ticket query
  $ticketQuery = "INSERT INTO `ticket` (`ticketCode`, `eventId`, `statusId`) VALUES (:ticketCode, :eventId, :statusId);";

  //add the event
  try {
    move_uploaded_file($_FILES['eventImg']['tmp_name'], $targetFilePath);

    $conn->beginTransaction();
    //event statement
    $stmtDetails = $conn->prepare($query);
```

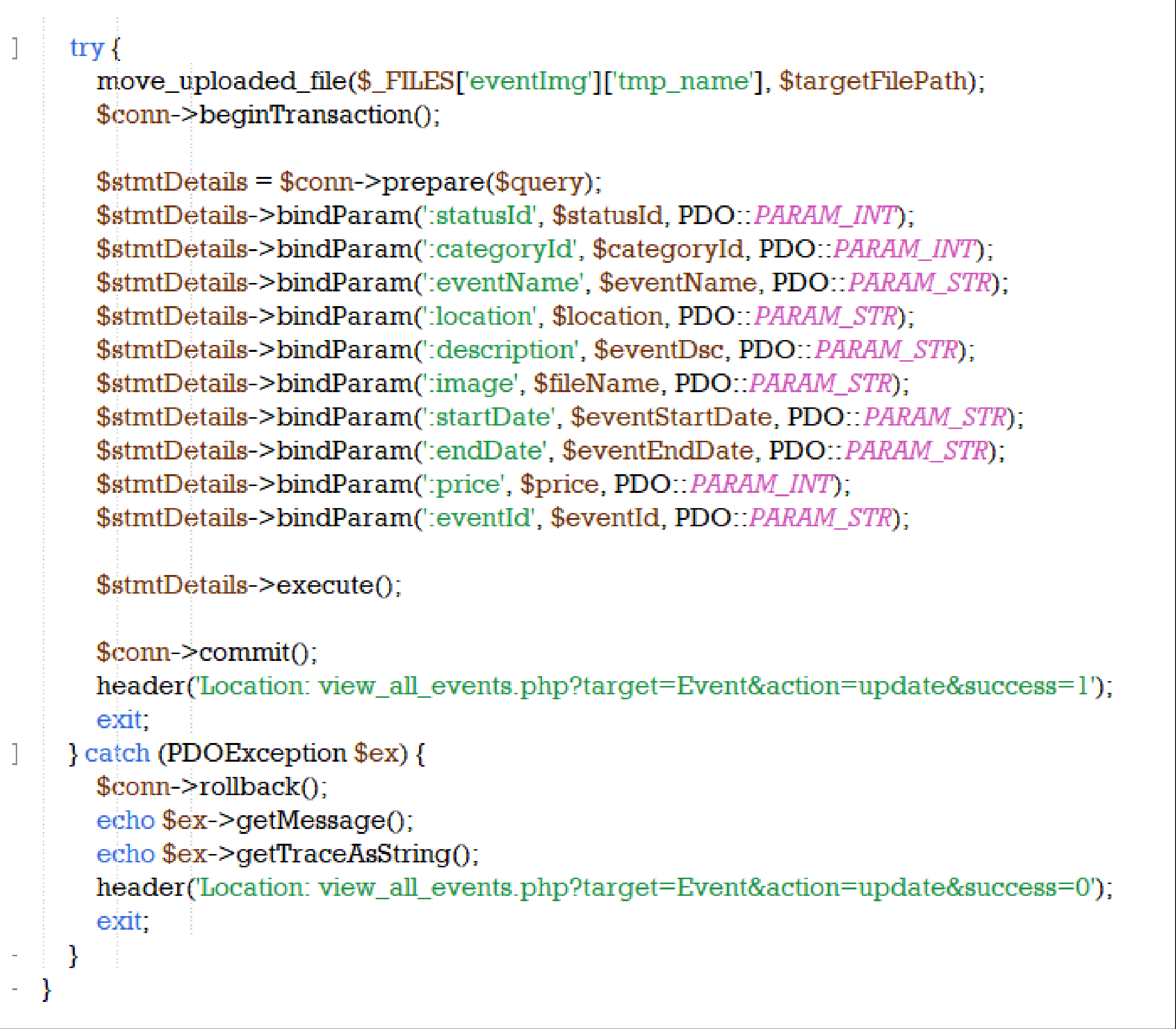


```
//binding for event
$stmtDetails->bindParam(':statusId', $statusId, PDO::PARAM_INT);
$stmtDetails->bindParam(':categoryId', $categoryId, PDO::PARAM_INT);
$stmtDetails->bindParam(':eventName', $eventName, PDO::PARAM_STR);
$stmtDetails->bindParam(':location', $location, PDO::PARAM_STR);
$stmtDetails->bindParam(':description', $eventDsc, PDO::PARAM_STR);
$stmtDetails->bindParam(':image', $fileName, PDO::PARAM_STR);
$stmtDetails->bindParam(':startDate', $eventStartDate, PDO::PARAM_STR);
$stmtDetails->bindParam(':endDate', $eventEndDate, PDO::PARAM_STR);
$stmtDetails->bindParam(':capacity', $capacity, PDO::PARAM_INT);
$stmtDetails->bindParam(':price', $price, PDO::PARAM_INT);
$stmtDetails->bindParam(':quantityLeft', $capacity, PDO::PARAM_INT);
$stmtDetails->bindParam(':createdBy', $createdBy, PDO::PARAM_STR);

$stmtDetails->execute();

//generate the tickets
//ticket statement
$newEventId = $conn->lastInsertId();
$ticketStmt = $conn->prepare($ticketQuery);
$ticketCodes = array();

$ticketStmt->bindParam(':eventId', $newEventId, PDO::PARAM_INT);
$ticketStmt->bindParam(':statusId', $statusId, PDO::PARAM_INT);

//bind the ticket code
for ($i = 0; $i < $capacity; $i++) {
  $ticketCode = generateTicketCode($eventName, 6); // Generate an 8-character ticket code
  $ticketStmt->bindParam(':ticketCode', $ticketCode, PDO::PARAM_STR);
  $ticketStmt->execute();
  $ticketCodes[] = $ticketCode; // Add the ticket code to the array
}

$conn->commit();
header('Location: view_all_events.php?target=Event&action=add&success=1');
} catch (Exception $ex) {
```

edit_event.php

- **Allow user to edit the event information and update it to the database**

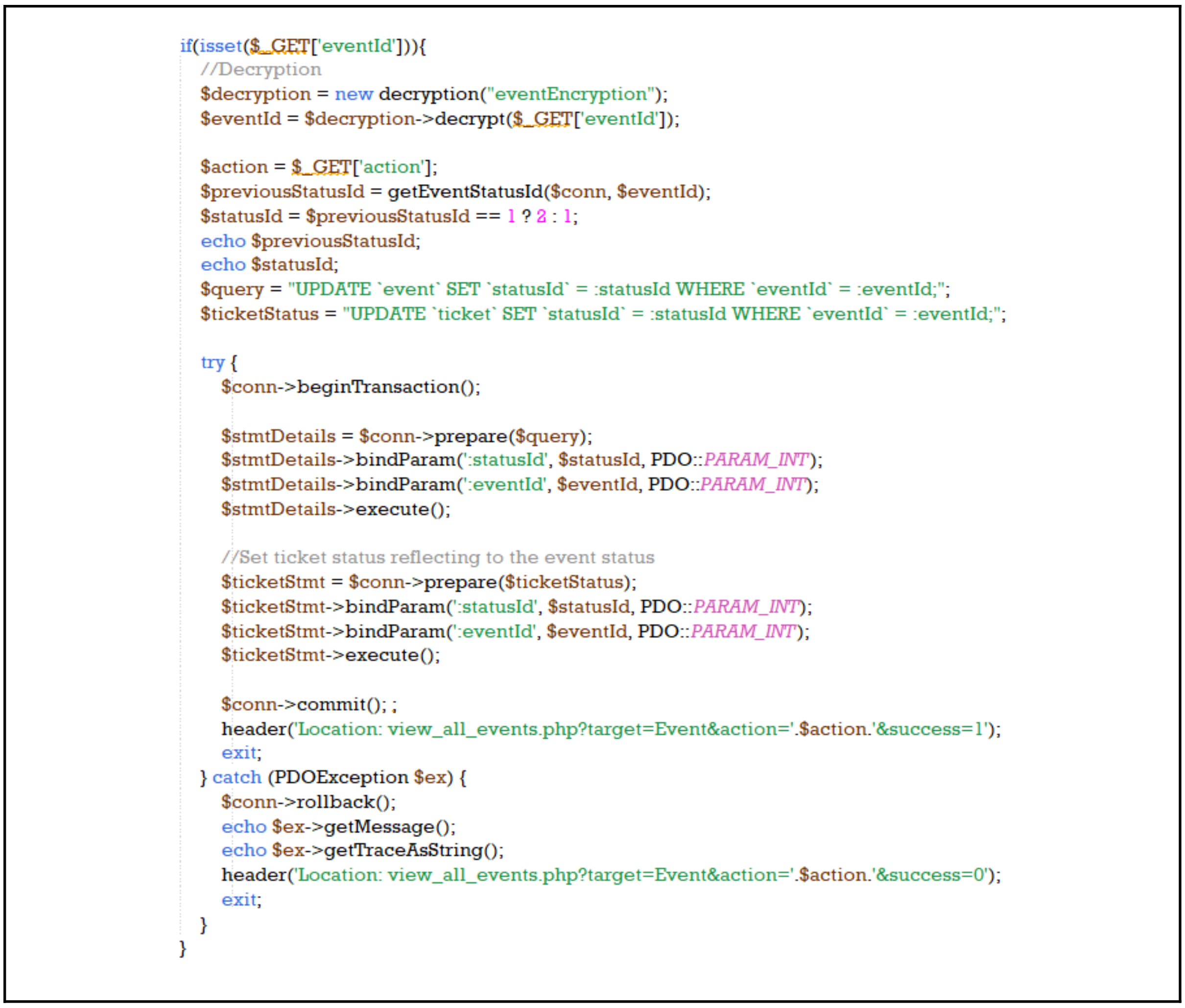
```
if (isset($_GET['eventId'])) {
    //Decryption
    $decryption = new decryption("eventEncryption");
    $eventId = $decryption->decrypt($_GET['eventId']);
    $eventData = getEventDataById($conn, $eventId);
    $displayStatus = getStatus($conn, $eventData['statusId']);
    $displayCategory = getCategories($conn, $eventData['categoryId']);
}


if (isset($_POST['editEvent'])) {
    $eventId = $_POST['eventId'];
    $eventName = $_POST['eventName'];
    $location = $_POST['location'];
    $eventDsc = $_POST['eventDsc'];
    $eventStartDate = $_POST['eventStartDate'];
    $eventEndDate = $_POST['eventEndDate'];
    $price = $_POST['price'];
    $categoryId = $_POST['category'];
    $statusId = $_POST['status'];

    //Get the input file name
    $fileName = isset($_FILES['eventImg']['name']) ? basename($_FILES['eventImg']['name']) : '';
    $targetDir = "../../pictures/";
    $targetFilePath = $targetDir . $fileName;
    $fileType = pathinfo($targetFilePath, PATHINFO_EXTENSION);

    $query = "UPDATE `event` SET `statusId` = :statusId,
         `categoryId` = :categoryId, `eventName` = :eventName,
         `location` = :location, `description` = :description,
         `image` = :image, `startDate` = :startDate,
         `endDate` = :endDate, `price` = :price
         WHERE `eventId` = :eventId";
```



```
    try {
        move_uploaded_file($_FILES['eventImg']['tmp_name'], $targetFilePath);
        $conn->beginTransaction();

        $stmtDetails = $conn->prepare($query);
        $stmtDetails->bindParam(':statusId', $statusId, PDO::PARAM_INT);
        $stmtDetails->bindParam(':categoryId', $categoryId, PDO::PARAM_INT);
        $stmtDetails->bindParam(':eventName', $eventName, PDO::PARAM_STR);
        $stmtDetails->bindParam(':location', $location, PDO::PARAM_STR);
        $stmtDetails->bindParam(':description', $eventDsc, PDO::PARAM_STR);
        $stmtDetails->bindParam(':image', $fileName, PDO::PARAM_STR);
        $stmtDetails->bindParam(':startDate', $eventStartDate, PDO::PARAM_STR);
        $stmtDetails->bindParam(':endDate', $eventEndDate, PDO::PARAM_STR);
        $stmtDetails->bindParam(':price', $price, PDO::PARAM_INT);
        $stmtDetails->bindParam(':eventId', $eventId, PDO::PARAM_STR);

        $stmtDetails->execute();

        $conn->commit();
        header('Location: view_all_events.php?target=Event&action=update&success=1');
        exit;
    } catch (PDOException $ex) {
        $conn->rollback();
        echo $ex->getMessage();
        echo $ex->getTraceAsString();
        header('Location: view_all_events.php?target=Event&action=update&success=0');
        exit;
    }
}
```

delete_recover_event.php

- **User have the option to delete and recover the event in this page**

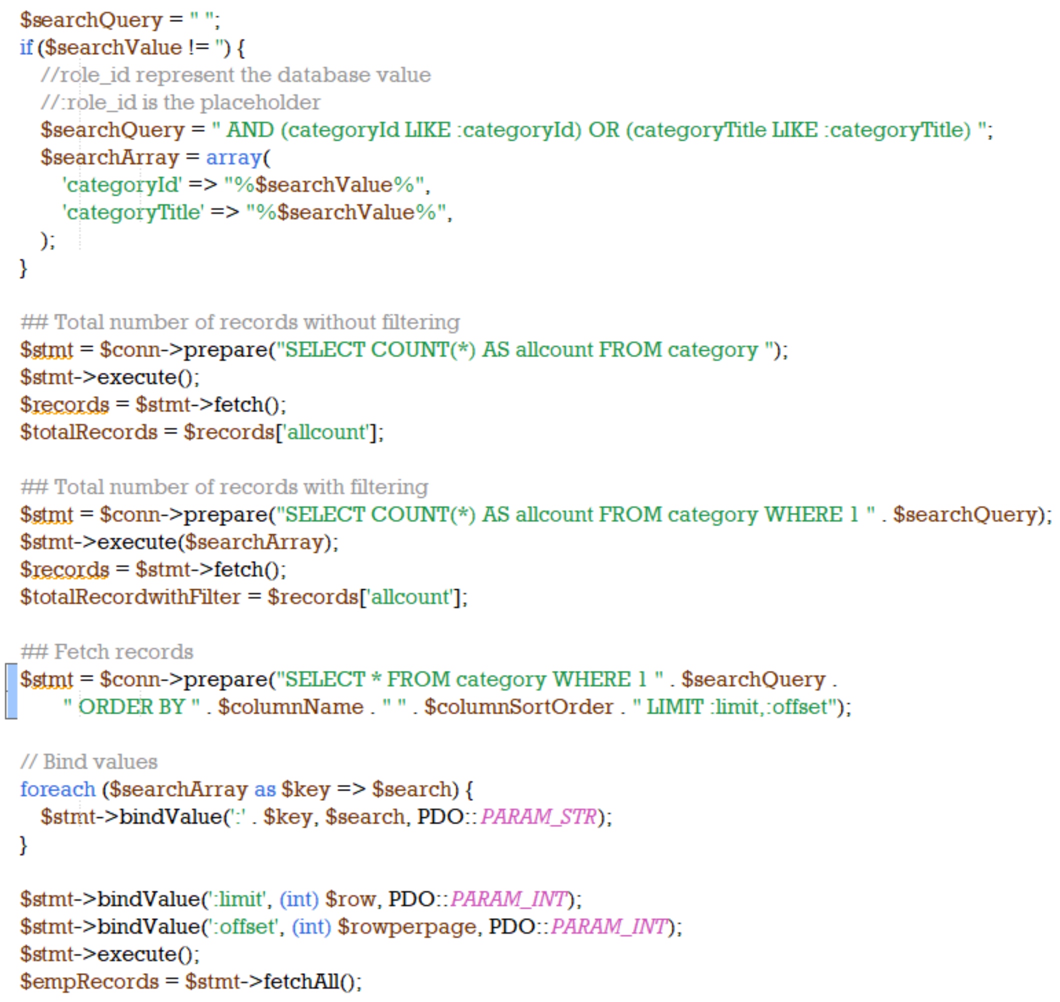


```php
if(isset($_GET['eventId'])){
    //Decryption
    $decryption = new decryption("eventEncryption");
    $eventId = $decryption->decrypt($_GET['eventId']);

    $action = $_GET['action'];
    $previousStatusId = getEventStatusId($conn, $eventId);
    $statusId = $previousStatusId == 1 ? 2 : 1;
    echo $previousStatusId;
    echo $statusId;
    $query = "UPDATE `event` SET `statusId` = :statusId WHERE `eventId` = :eventId;";
    $ticketStatus = "UPDATE `ticket` SET `statusId` = :statusId WHERE `eventId` = :eventId;";

    try {
        $conn->beginTransaction();

        $stmtDetails = $conn->prepare($query);
        $stmtDetails->bindParam(':statusId', $statusId, PDO::PARAM_INT);
        $stmtDetails->bindParam(':eventId', $eventId, PDO::PARAM_INT);
        $stmtDetails->execute();

        //Set ticket status reflecting to the event status
        $ticketStmt = $conn->prepare($ticketStatus);
        $ticketStmt->bindParam(':statusId', $statusId, PDO::PARAM_INT);
        $ticketStmt->bindParam(':eventId', $eventId, PDO::PARAM_INT);
        $ticketStmt->execute();

        $conn->commit(); ;
        header('Location: view_all_events.php?target=Event&action='.$action.'&success=1');
        exit;
    } catch (PDOException $ex) {
        $conn->rollback();
        echo $ex->getMessage();
        echo $ex->getTraceAsString();
        header('Location: view_all_events.php?target=Event&action='.$action.'&success=0');
        exit;
    }
}
```

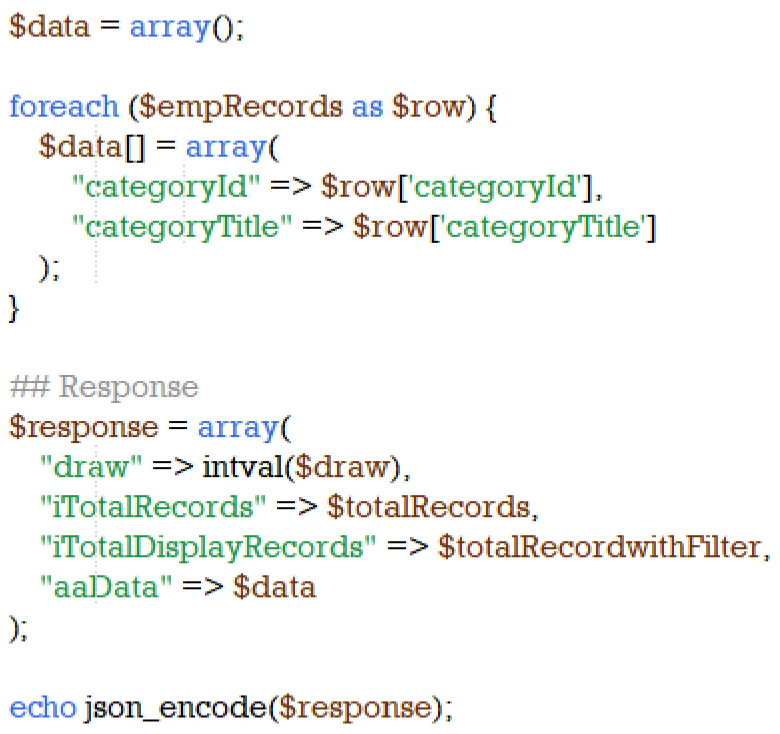
3.2.3 Category Module

categories_view.php

- **Retrieve the category data from database and process them into Datatable format**
- **Apply pagination, searching, filtering and sorting on the data**



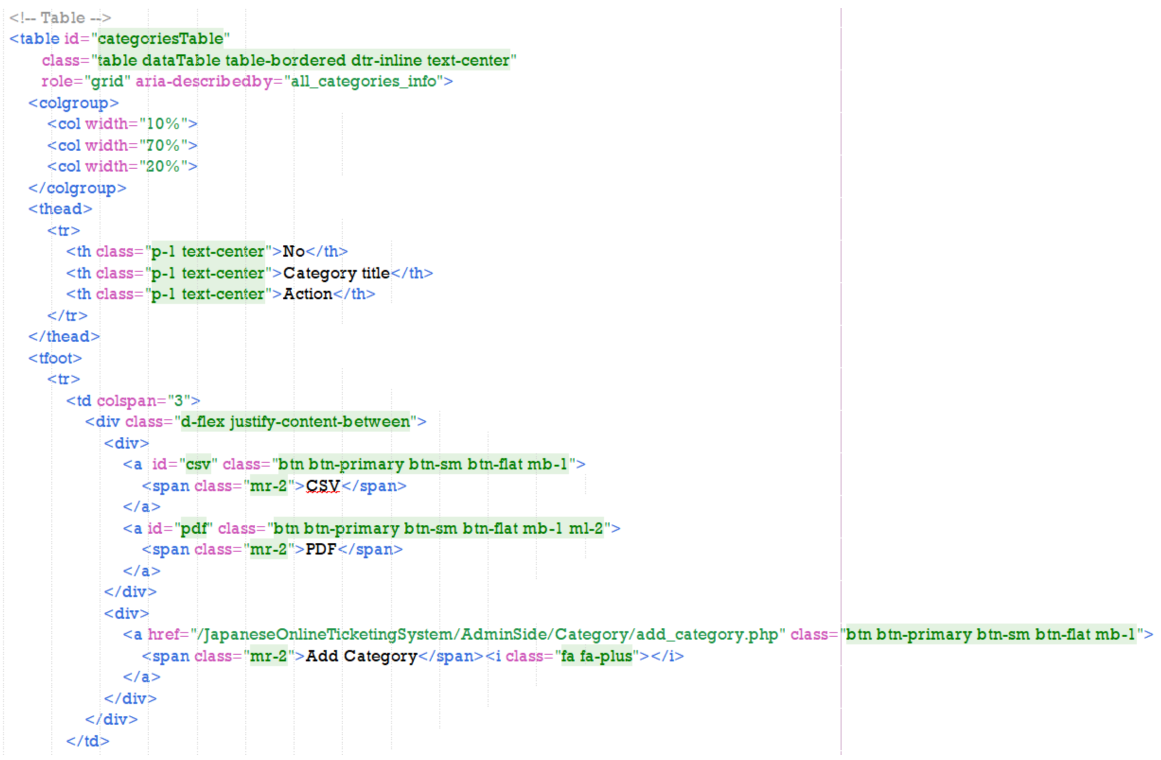
```php
$searchQuery = " ";
if ($searchValue != '') {
    //role_id represent the database value
    //:role_id is the placeholder
    $searchQuery = " AND (categoryId LIKE :categoryId) OR (categoryTitle LIKE :categoryTitle) ";
    $searchArray = array(
        'categoryId' => "%$searchValue%",
        'categoryTitle' => "%$searchValue%",
    );
}

## Total number of records without filtering
$stmt = $conn->prepare("SELECT COUNT(*) AS allcount FROM category ");
$stmt->execute();
$records = $stmt->fetch();
$totalRecords = $records['allcount'];

## Total number of records with filtering
$stmt = $conn->prepare("SELECT COUNT(*) AS allcount FROM category WHERE 1 " . $searchQuery);
$stmt->execute($searchArray);
$records = $stmt->fetch();
$totalRecordwithFilter = $records['allcount'];

## Fetch records
$stmt = $conn->prepare("SELECT * FROM category WHERE 1 " . $searchQuery .
    " ORDER BY " . $columnName . " " . $columnSortOrder . " LIMIT :limit,:offset");

// Bind values
foreach ($searchArray as $key => $search) {
    $stmt->bindValue(':' . $key, $search, PDO::PARAM_STR);
}

$stmt->bindValue(':limit', (int) $row, PDO::PARAM_INT);
$stmt->bindValue(':offset', (int) $rowperpage, PDO::PARAM_INT);
$stmt->execute();
$empRecords = $stmt->fetchAll();
```



```php
$data = array();

foreach ($empRecords as $row) {
   $data[] = array(
      "categoryId" => $row['categoryId'],
      "categoryTitle" => $row['categoryTitle']
   );
}

## Response
$response = array(
   "draw" => intval($draw),
   "iTotalRecords" => $totalRecords,
   "iTotalDisplayRecords" => $totalRecordwithFilter,
   "aaData" => $data
);

echo json_encode($response);
```

view_all_categories.php

- **Display the category Datatable to the user**



```html
<!-- Table -->
<table id="categoriesTable"
   class="table dataTable table-bordered dtr-inline text-center"
   role="grid" aria-describedby="all_categories_info">
  <colgroup>
    <col width="10%">
    <col width="70%">
    <col width="20%">
  </colgroup>
  <thead>
    <tr>
      <th class="p-1 text-center">No</th>
      <th class="p-1 text-center">Category title</th>
      <th class="p-1 text-center">Action</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="3">
        <div class="d-flex justify-content-between">
          <div>
            <a  id="csv" class="btn btn-primary btn-sm btn-flat mb-1">
              <span class="mr-2">CSV</span>
            </a>
            <a id="pdf" class="btn btn-primary btn-sm btn-flat mb-1 ml-2">
              <span class="mr-2">PDF</span>
            </a>
          </div>
          <div>
            <a href="/JapaneseOnlineTicketingSystem/AdminSide/Category/add_category.php" class="btn btn-primary btn-sm btn-flat mb-1">
              <span class="mr-2">Add Category</span><i class="fa fa-plus"></i>
            </a>
          </div>
        </div>
      </td>
```

view_category_jquery.js

- **Retrieve the category Datatabe from the server side and render the data to the html table**

```
$('#categoriesTable').DataTable({
  'processing': true,
  'serverSide': true,
  'serverMethod': 'post',
  'ajax': {
    'url': '/JapaneseOnlineTicketingSystem/dataTable/categories_view.php'
  },
  'columns': [
    {data: 'categoryId'},
    {data: 'categoryTitle'},
    {
      render: function (data, type, row, meta) {
        //Encryption
        let encryptedResult = JSON.parse(encryptData(row.categoryId));
        let encryptedCategoryId = encryptedResult.encryptedCategoryId;

        var html = `
                <a class="btn btn-primary btn-floating" title="Edit" href="edit_category.php?categoryId=${encryptedCategoryId}" role="button">
                  <i class="fas fa fa-pen"></i>
                </a>
                `;
        return html;
      },
      orderable: false
    }
  ]
});
```

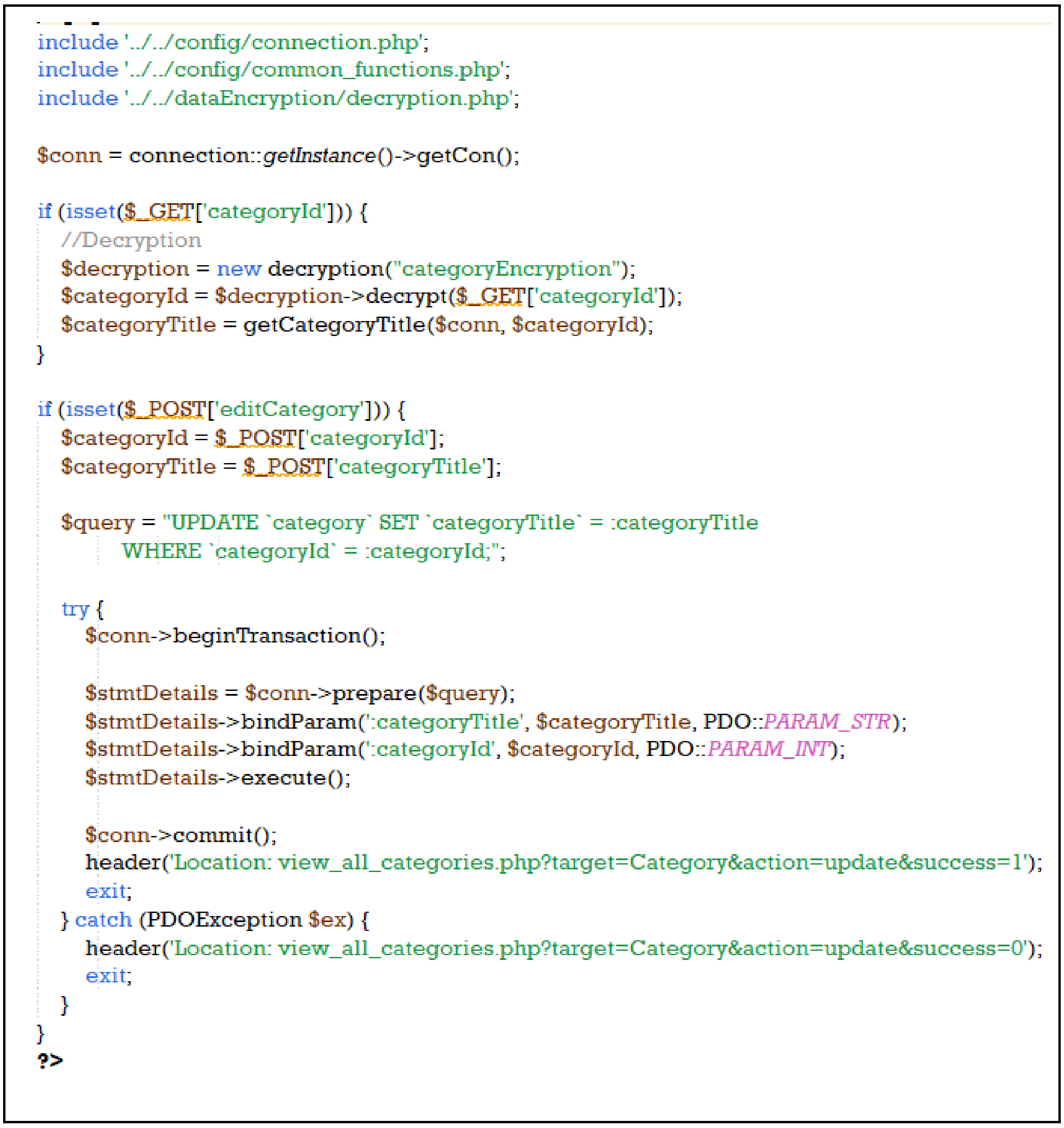
add_category.php

- **Allow user to add category**

```
require_once '../../config/connection.php';

$conn = connection::getInstance()->getCon();

if (isset($_POST['addCategory'])) {
  $categoryTitle = $_POST['categoryTitle'];

  $query = "INSERT INTO `category` (`categoryTitle`)
        VALUES (:categoryTitle);";

  try {
    $conn->beginTransaction();
    $stmtDetails = $conn->prepare($query);
    $stmtDetails->bindParam(':categoryTitle', $categoryTitle, PDO::PARAM_STR);
    $stmtDetails->execute();

    $conn->commit();
    header('Location: view_all_categories.php?target=Category&action=add&success=1');
    exit;
  } catch (PDOException $ex) {
    header('Location: view_all_categories.php?target=Category&action=add&success=0');
    exit;
  }
}
?>
```

edit_category.php

- **Allow user to update the category**



```php
include '../../config/connection.php';
include '../../config/common_functions.php';
include '../../dataEncryption/decryption.php';

$conn = connection::getInstance()->getCon();

if (isset($_GET['categoryId'])) {
    //Decryption
    $decryption = new decryption("categoryEncryption");
    $categoryId = $decryption->decrypt($_GET['categoryId']);
    $categoryTitle = getCategoryTitle($conn, $categoryId);
}

if (isset($_POST['editCategory'])) {
    $categoryId = $_POST['categoryId'];
    $categoryTitle = $_POST['categoryTitle'];

    $query = "UPDATE `category` SET `categoryTitle` = :categoryTitle
            WHERE `categoryId` = :categoryId;";

    try {
        $conn->beginTransaction();

        $stmtDetails = $conn->prepare($query);
        $stmtDetails->bindParam(':categoryTitle', $categoryTitle, PDO::PARAM_STR);
        $stmtDetails->bindParam(':categoryId', $categoryId, PDO::PARAM_INT);
        $stmtDetails->execute();

        $conn->commit();
        header('Location: view_all_categories.php?target=Category&action=update&success=1');
        exit;
    } catch (PDOException $ex) {
        header('Location: view_all_categories.php?target=Category&action=update&success=0');
        exit;
    }
}
?>
```

4. Design Pattern
In our web project, I choose to use the **virtual proxy** pattern.

4.1 Description of Design Pattern
Virtual proxy pattern belongs to the Proxy pattern family which is a type of structural design pattern. It involves the use of the proxy class to represent the real object, which may be expensive to create or access. The proxy class provides a placeholder for the real object, and delays its creation or access until it is actually needed. The common structure for virtual proxy includes a subject interface, a real subject class, a proxy class and a client that uses the proxy class to access the real object.

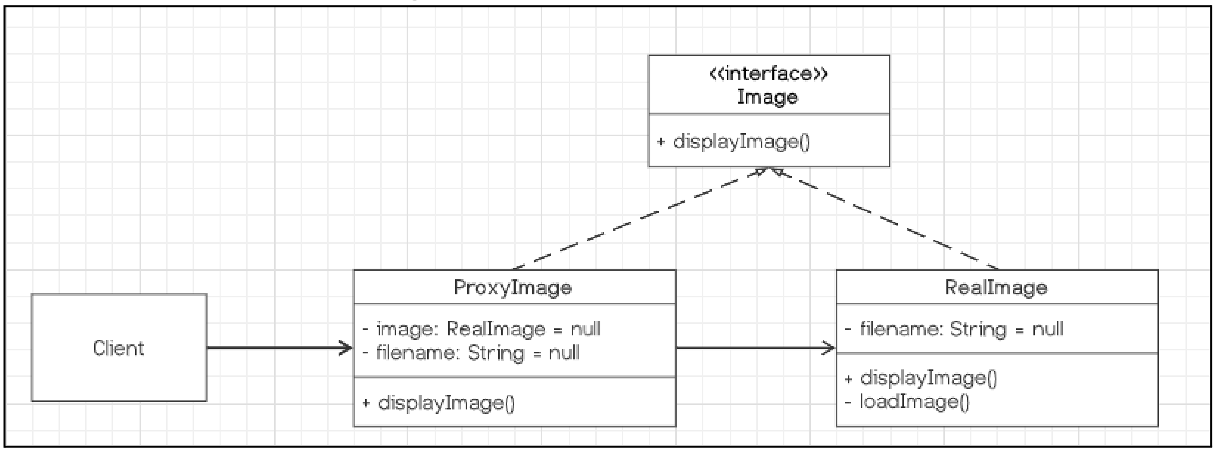
4.2 Implementation of Design Pattern


Figure 1.0: Virtual Proxy Class Diagram

I have applied the virtual proxy pattern on client side pages such as home page and event page. The main reason I choose to use the virtual proxy pattern is because it provides **performance optimization**. In the case of image loading, it can be particularly useful as loading large images can consume a lot of resources and time especially in our web application which has a lot of images to display. By using a proxy pattern, it is able to avoid the loading and manipulation of large images until they are actually required which results in a better performance. Additionally, due to the virtual proxy pattern able to cache the objects once it has been created or loaded, it is able to **reduce the application workload** by displaying the subsequent for the same object directly from the cache instead of loading the object again. To illustrate this, a more detailed explanation of how this proxy pattern works is shown below.

The ProxyImage class is a virtual proxy that represents the RealImage object. When the displayImage() method of the ProxyImage object is called, it checks whether the RealImage object has been created yet. If not, it creates a new RealImage object and loads the image.

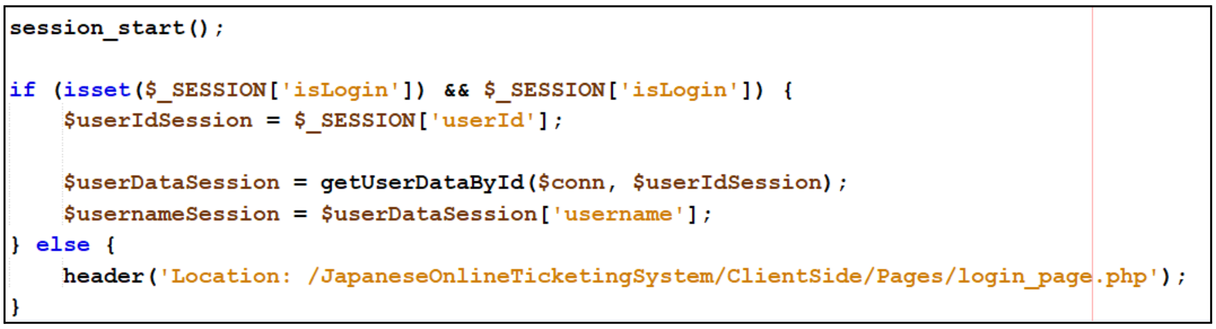
If the RealImage object has already been created, it simply displays the image. In other words, it just requires one load for every image.

This can clearly indicate how the virtual proxy pattern can be used to defer the creation of expensive objects until they are actually needed, and how it can improve the performance of a system by reducing the number of objects that are created and managed. Overall, the proxy pattern is being used to optimize the loading of images and reduce server load by caching the images.

5. Software Security

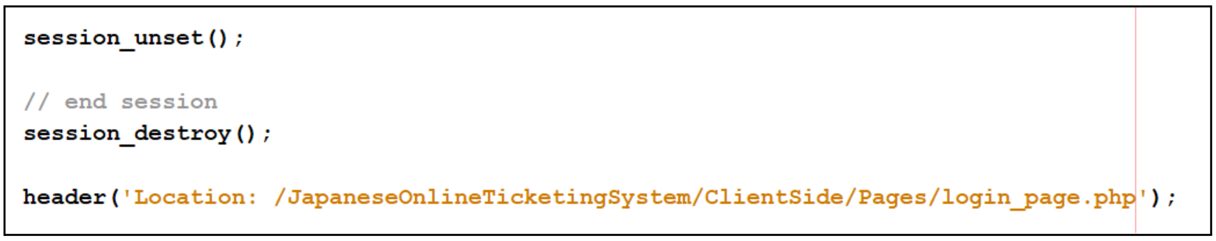
5.1 Secure Coding Practices

5.1.1 Session Management



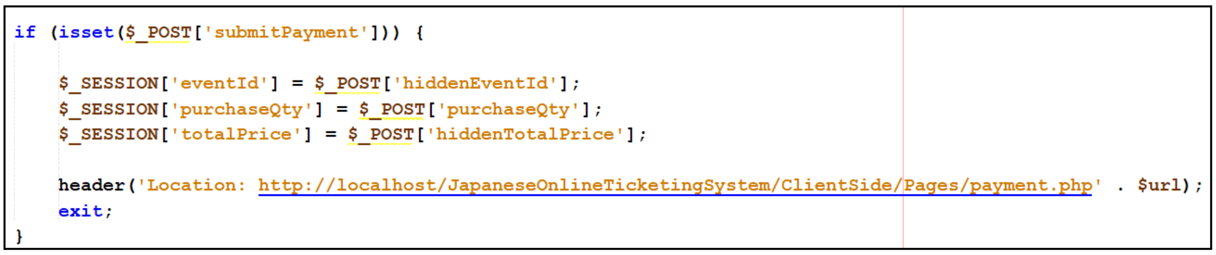
```
session_start();

if (isset($_SESSION['isLogin']) && $_SESSION['isLogin']) {
    $userIdSession = $_SESSION['userId'];

    $userDataSession = getUserDataById($conn, $userIdSession);
    $usernameSession = $userDataSession['username'];
} else {
    header('Location: /JapaneseOnlineTicketingSystem/ClientSide/Pages/login_page.php');
}
```

The first security feature has been implemented is session management in the login function. Above code is used to store the user details into session when they successfully login by entering correct credentials. In this case, user id and its name is being stored in the session variable so that the application is able to know currently who is the user that is visiting the website.

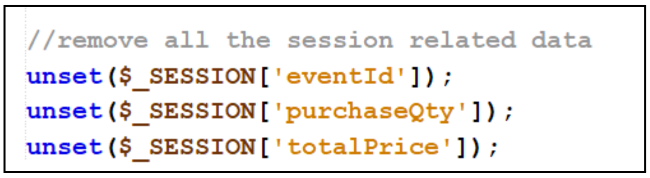


```
session_unset();

// end session
session_destroy();

header('Location: /JapaneseOnlineTicketingSystem/ClientSide/Pages/login_page.php');
```

Moreover, when a user logout from their account, the data in session will be removed by using the unset and destroy function. This helps to remove sensitive data from the session, reducing the risk of data leakage or unauthorized access to the information in subsequent requests.



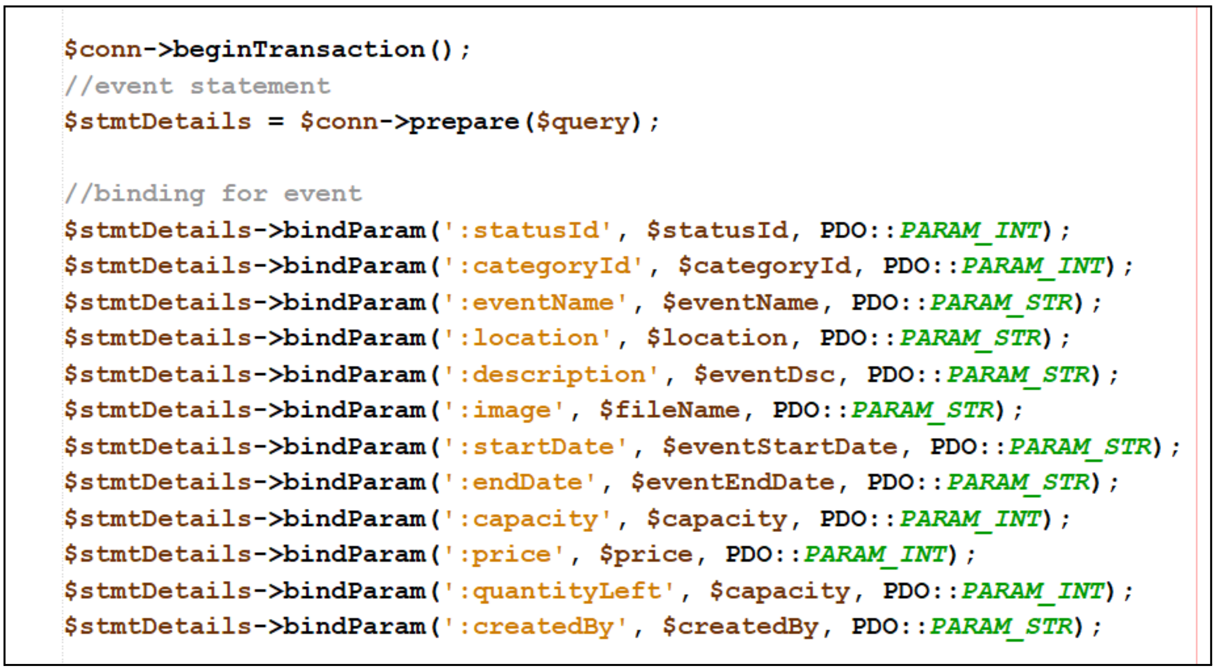
```
if (isset($_POST['submitPayment'])) {

    $_SESSION['eventId'] = $_POST['hiddenEventId'];
    $_SESSION['purchaseQty'] = $_POST['purchaseQty'];
    $_SESSION['totalPrice'] = $_POST['hiddenTotalPrice'];

    header('Location: http://localhost/JapaneseOnlineTicketingSystem/ClientSide/Pages/payment.php' . $url);
    exit;
}
```



```
//remove all the session related data
unset($_SESSION['eventId']);
unset($_SESSION['purchaseQty']);
unset($_SESSION['totalPrice']);
```

Other than the login function, I have also implemented the session management in the event page where it will store the event and ticket data into session. Then, the session variables are going to be cleared after the user makes a successful payment.

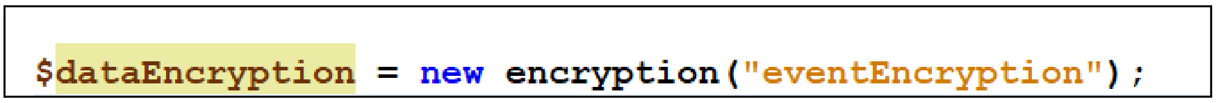
5.1.2 Database Security



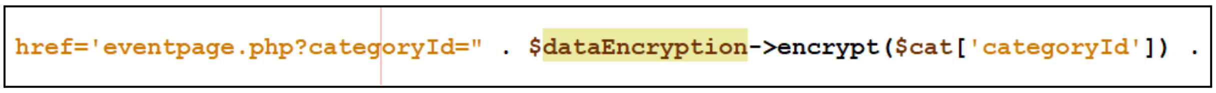
```php
$conn->beginTransaction();
//event statement
$stmtDetails = $conn->prepare($query);

//binding for event
$stmtDetails->bindParam(':statusId', $statusId, PDO::PARAM_INT);
$stmtDetails->bindParam(':categoryId', $categoryId, PDO::PARAM_INT);
$stmtDetails->bindParam(':eventName', $eventName, PDO::PARAM_STR);
$stmtDetails->bindParam(':location', $location, PDO::PARAM_STR);
$stmtDetails->bindParam(':description', $eventDsc, PDO::PARAM_STR);
$stmtDetails->bindParam(':image', $fileName, PDO::PARAM_STR);
$stmtDetails->bindParam(':startDate', $eventStartDate, PDO::PARAM_STR);
$stmtDetails->bindParam(':endDate', $eventEndDate, PDO::PARAM_STR);
$stmtDetails->bindParam(':capacity', $capacity, PDO::PARAM_INT);
$stmtDetails->bindParam(':price', $price, PDO::PARAM_INT);
$stmtDetails->bindParam(':quantityLeft', $capacity, PDO::PARAM_INT);
$stmtDetails->bindParam(':createdBy', $createdBy, PDO::PARAM_STR);
```

Second security feature is database security. This is being implemented for all the sql queries. I ensure that there are no hardcodes in the sql query instead I uses PDO prepared statement to pass the variable into the sql. This is because using parameterized query and prepared statements is able to protect against SQL injection.

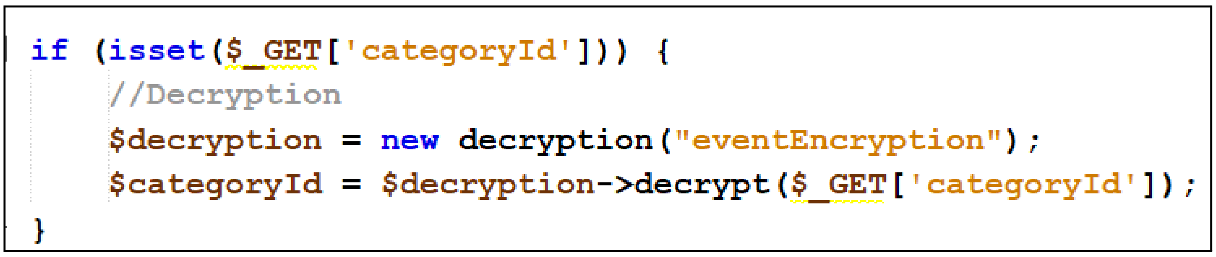
5.1.3 Data Protection



```php
$dataEncryption = new encryption("eventEncryption");
```

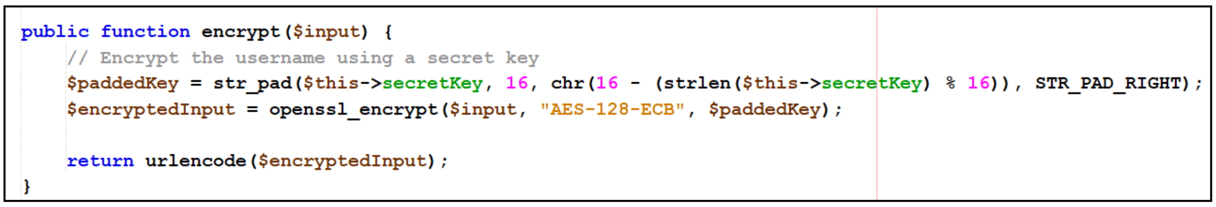


```php
href='eventpage.php?categoryId=" . $dataEncryption->encrypt($cat['categoryId']) .
```

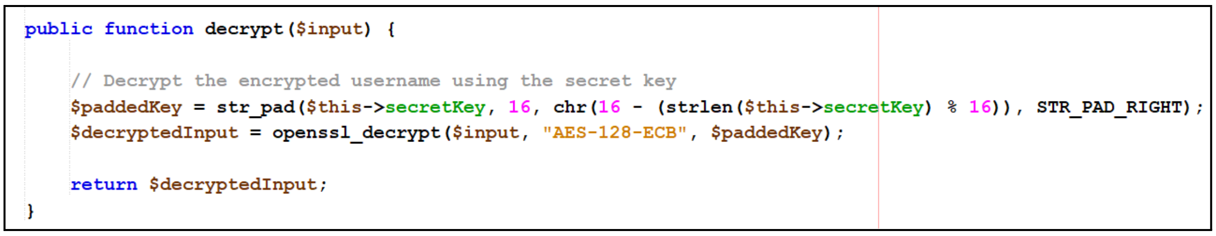


```php
if (isset($_GET['categoryId'])) {
    //Decryption
    $decryption = new decryption("eventEncryption");
    $categoryId = $decryption->decrypt($_GET['categoryId']);
}
```

The third security feature is data protection. This security feature is also being implemented for all the pages that use the GET method to retrieve data from the url. I have encrypted all the data in the url and provided a shared key as shown in the screenshot above (first and second screenshot). The shared key named “eventEncryption” can only be decrypted by using the same shared key which means if the decryption has a different name of shared key, the data would not be able to decrypt (third screenshot).



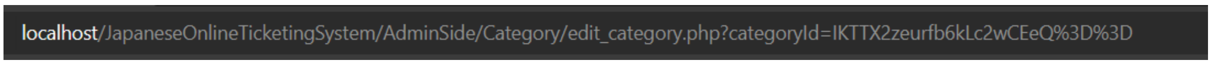
```
public function encrypt($input) {
    // Encrypt the username using a secret key
    $paddedKey = str_pad($this->secretKey, 16, chr(16 - (strlen($this->secretKey) % 16)), STR_PAD_RIGHT);
    $encryptedInput = openssl_encrypt($input, "AES-128-ECB", $paddedKey);

    return urlencode($encryptedInput);
}
```



```
public function decrypt($input) {

    // Decrypt the encrypted username using the secret key
    $paddedKey = str_pad($this->secretKey, 16, chr(16 - (strlen($this->secretKey) % 16)), STR_PAD_RIGHT);
    $decryptedInput = openssl_decrypt($input, "AES-128-ECB", $paddedKey);

    return $decryptedInput;
}
```

Above screenshots are the details of the encrypt and decrypt function. The decrypt function should use the same method as the encrypt function otherwise the data would also not be able to decrypt. The actual encryption and decryption is the openssl_encrypt and decrypt keywords, and the $paddedKey is just to format the data needed to be encrypt and decrypt. The function urlencode is also being applied to encrypt the encrypted data and safely transmit over the url.

Screenshots:



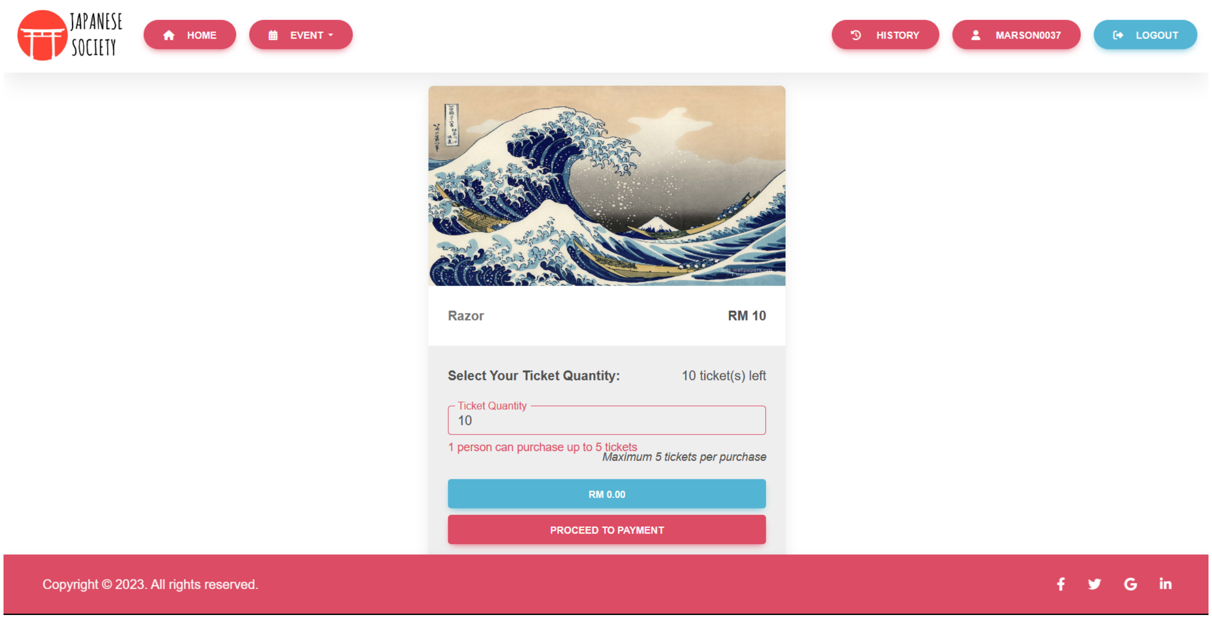
```
localhost/JapaneseOnlineTicketingSystem/AdminSide/Category/edit_category.php?categoryId=IKTTX2zeurfb6kLc2wCEeQ%3D%3D
```

Above screenshot shows the categoryId has been encrypted which does not allow the user to see its actual data and value.

5.1.4 File Management

Below will has the screenshot which shows the file management. Only specific formats can be uploaded.

5.1.5 Input Validation



Razor RM 10

Select Your Ticket Quantity: 10 ticket(s) left

Ticket Quantity: 10

1 person can purchase up to 5 tickets

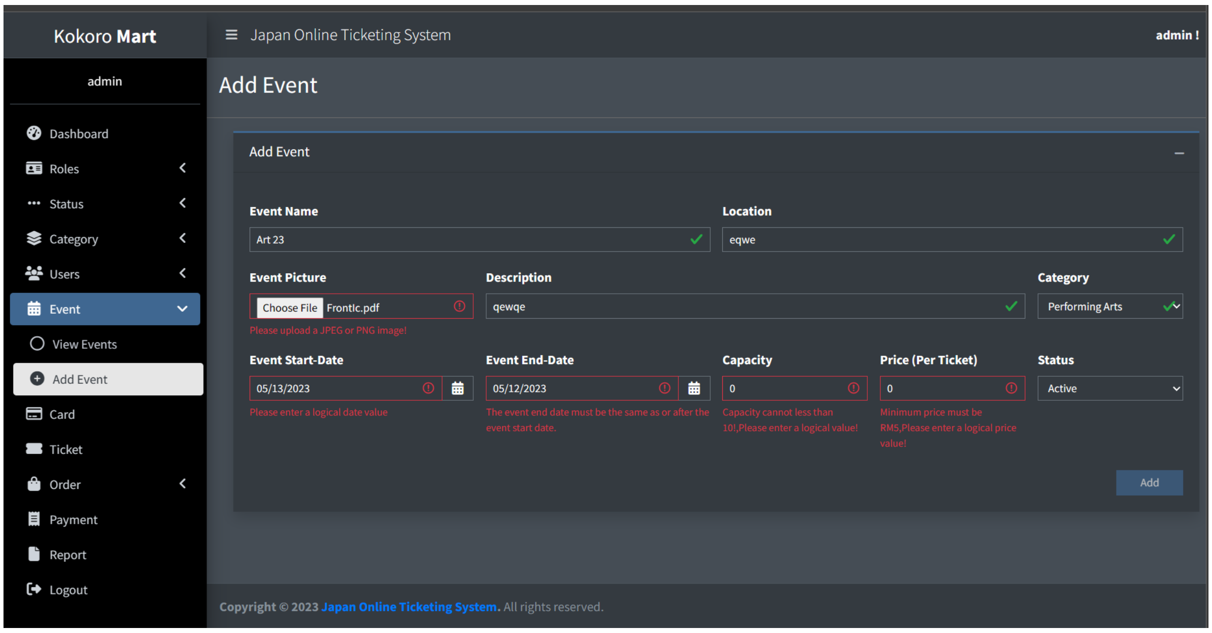
Maximum 5 tickets per purchase

RM 0.00

PROCEED TO PAYMENT

Copyright © 2023. All rights reserved.

Above screenshot shows that the application will have validation checking on the ticket quantity selected by the user. If users try to select more than 5 tickets or the amount of tickets which are more than the available ticket, it is not allowed.

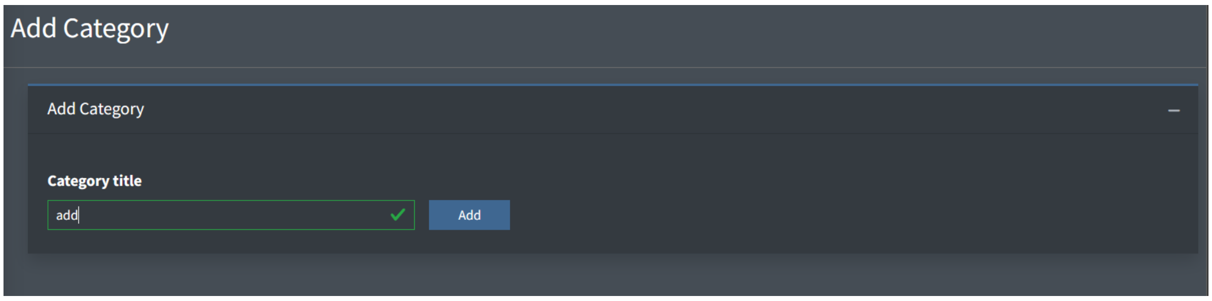


Add Event

Event Name	Art 23
Location	eqwe
Event Picture	Choose File Frontlc.pdf — Please upload a JPEG or PNG image!
Description	qewqe
Category	Performing Arts
Event Start-Date	05/13/2023 — Please enter a logical date value
Event End-Date	05/12/2023 — The event end date must be the same as or after the event start date.
Capacity	0 — Capacity cannot less than 10!,Please enter a logical value!
Price (Per Ticket)	0 — Minimum price must be RM5,Please enter a logical price value!
Status	Active

Copyright © 2023 Japan Online Ticketing System. All rights reserved.

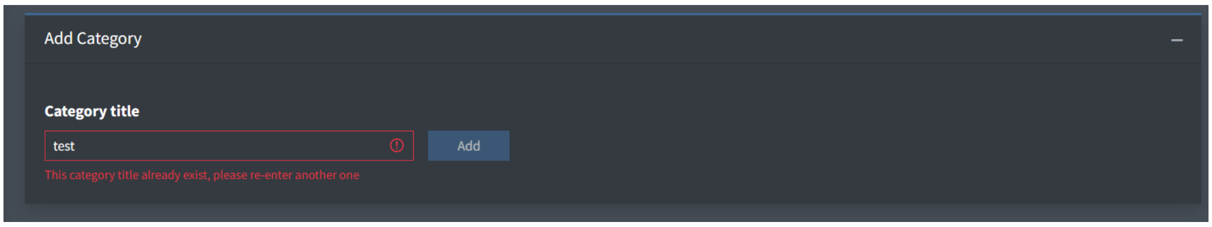
Above screenshot shows the validation check when staff try to add and upload a new event. The picture format has been restricted to only JPEG and PNG format. The event start data should also be one week after the current date, and date before today is not allowed. It is the same as the end date as it must be after the start date. Capacity and price should also be at least 10.


Add Category

Add Category

Category title

add


Add Category

Category title

test

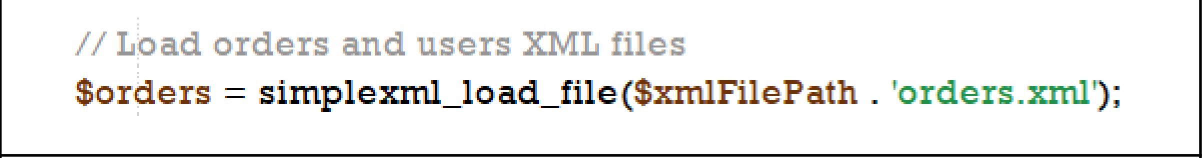
This category title already exist, please re-enter another one

Above screenshots showing that duplicate category title is not allowed.

6. Implementation of XML, XSLT and XPath

6.1 Overview (Top 5 most potential user report)

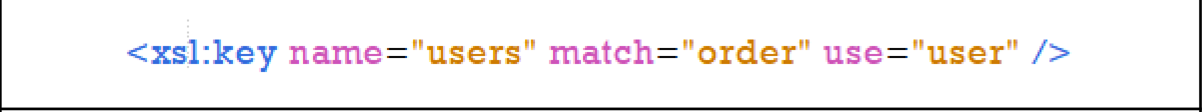
XML, XPATH and XLTS are used to create the top potential user reports. First, XML is used to initialize and create several xml files that contain the necessary information needed for reporting. Then, an XLTS file that uses XPATH to define the data xml format to be displayed will be created. Afterwards, a new XLTS processor will be created and used to import the created XLTS file stylesheet. The XLTS processor converts the results of the XPath query into pdf format. The pdf will be displayed to the user and can be downloaded.



```
// Load orders and users XML files
$orders = simplexml_load_file($xmlFilePath . 'orders.xml');
```

Step 1:

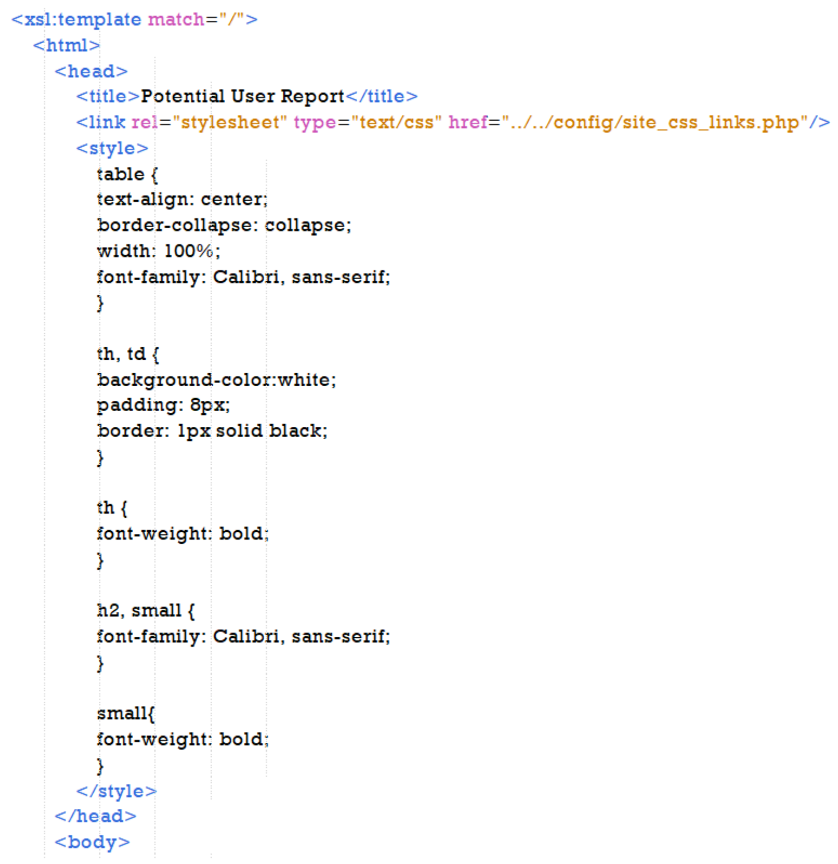
- First we start by loading the XML data file that will be used as the data for the report.



```
<xsl:key name="users" match="order" use="user" />
```

Step 2:

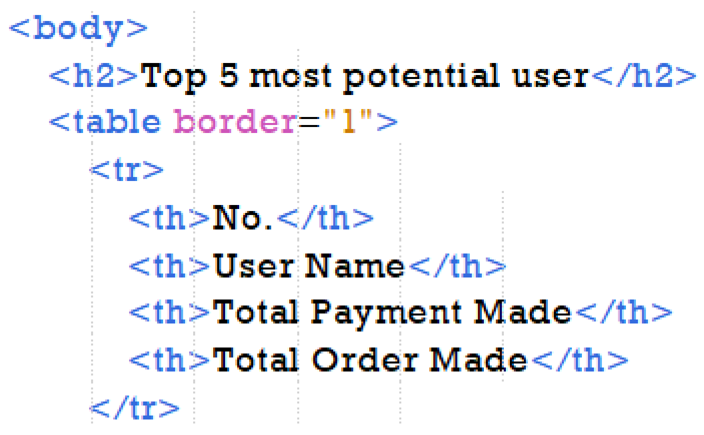
- Now we start to create the XLS file which uses the XLTS language and XPATH conditions to process the XML data into the required pdf report format.
- Before creating the XLTS template, we need to create an XLTS key first.
    - The key will have a name that allows it to be referenced throughout the XLTS stylesheet.
    - The match will find all the order tag elements in the input XML document, and the user specified which element under the order element will be used as the key value.



```
<xsl:template match="/">
  <html>
    <head>
      <title>Potential User Report</title>
      <link rel="stylesheet" type="text/css" href="../../config/site_css_links.php"/>
      <style>
        table {
        text-align: center;
        border-collapse: collapse;
        width: 100%;
        font-family: Calibri, sans-serif;
        }

        th, td {
        background-color:white;
        padding: 8px;
        border: 1px solid black;
        }

        th {
        font-weight: bold;
        }

        h2, small {
        font-family: Calibri, sans-serif;
        }

        small{
        font-weight: bold;
        }
      </style>
    </head>
    <body>
```

Step 3:

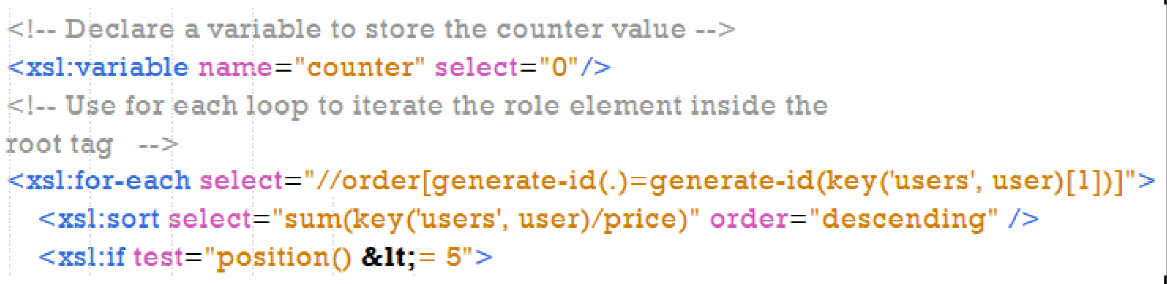
- Next, we create the XLTS template that will match the root node in the XML file
- Then inside the template, we give the report pdf page a html title.
- After that, we define the CSS styles that will be used to convert the html table to pdf table format.



```
<body>
  <h2>Top 5 most potential user</h2>
  <table border="1">
    <tr>
      <th>No.</th>
      <th>User Name</th>
      <th>Total Payment Made</th>
      <th>Total Order Made</th>
    </tr>
```

Step 4:

- Next, declare the heading for the report table, in this case, the report is titled as “Top 5 most potential users”.
- Following we create the html table that will be used to display the report data.
  - In this table, we first declare the 4 column headers, which are “No”, “User Name”, “Total Payment Made” and “Total Order Made”.
  - The “No.” header will be used to label the ranking of each user in a specified order.
  - Meanwhile the other 3 headers label the data that will be generated and viewed by the manager, or admin.

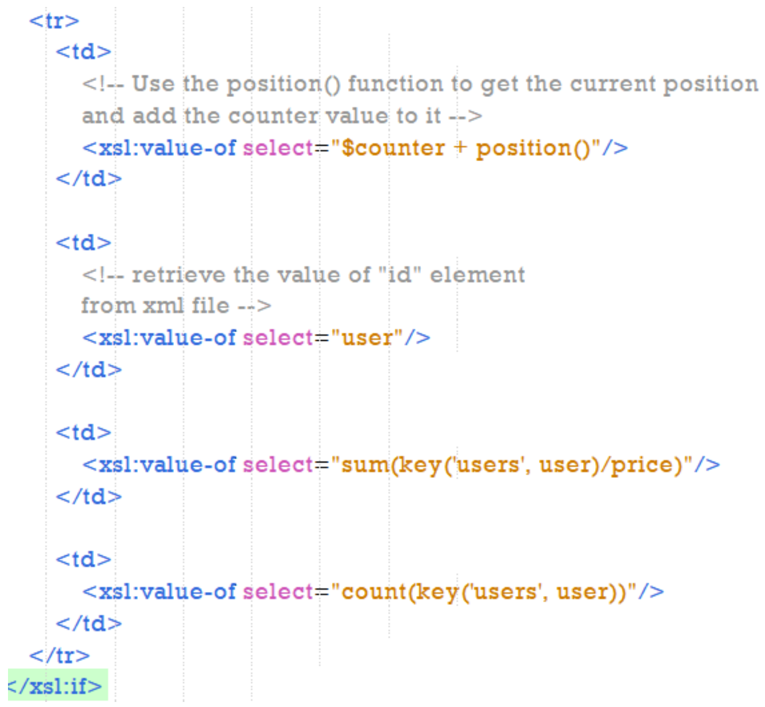


```
<!-- Declare a variable to store the counter value -->
<xsl:variable name="counter" select="0"/>
<!-- Use for each loop to iterate the role element inside the
root tag  -->
<xsl:for-each select="//order[generate-id(.)=generate-id(key('users', user)[1])]">
  <xsl:sort select="sum(key('users', user)/price)" order="descending" />
  <xsl:if test="position() &lt;= 5">
```

Step 5:

- Next, we create an XLTS variable “counter” to count the number of records in the report.
  - By default, the value of this counter will be 0.
  - It is also used to represent the rank of each record.
- Afterwards, we will start iterating by using the XLTS for-each loop.
  - A XPATH condition is used for this for-each loop. In this XPATH condition, we will be using the generate-id function and the key value that have been defined in the early stage.
  - In this case, the key-value will be used to select the first occurrence of a unique username under the order label
  - The generate-id function is used to generate an unique identifier key for each unique username that is selected by the key. Then, the expression “=” used to check whether the current node is the first occurrence of the node under the key group.
    - This is done by comparing the generated ID to the ID of the group
    - By doing so, it can get rid of the problem of repeated users in the report.

- After the for-each loop, following is the sorting of the data.
    - Sorting is also based on XPATH conditions. It will sort the sum of all prices of the same user group under the order element in descending order
- After the sort is the XLTS if function that will be used to limit the display of the report data to only 5 records.
    - In this case, the position() feature will return the current position of the current user group node. If the returned position is 5, the if operation will be ended.
    - *the &lt equals the “<” symbol.



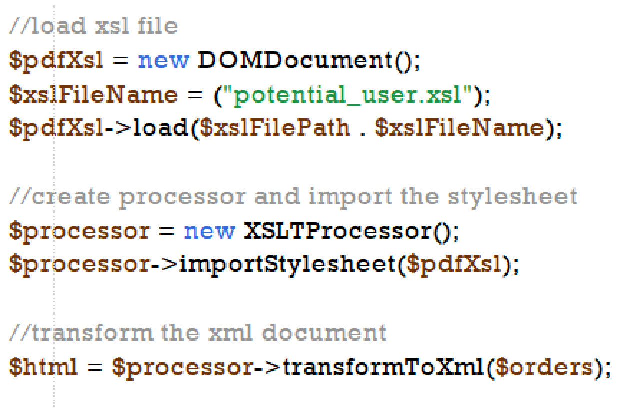
```
  <tr>
    <td>
      <!-- Use the position() function to get the current position
      and add the counter value to it -->
      <xsl:value-of select="$counter + position()"/>
    </td>

    <td>
      <!-- retrieve the value of "id" element
      from xml file -->
      <xsl:value-of select="user"/>
    </td>

    <td>
      <xsl:value-of select="sum(key('users', user)/price)"/>
    </td>

    <td>
      <xsl:value-of select="count(key('users', user))"/>
    </td>
  </tr>
</xsl:if>
```

Step 6:

- Now move on to the html table row and cell.
    - For the first cell, it will be used to display the “No.” value that labels the ranking of the record.
        - It uses the value of the defined variable $counter to sum up with the current position of the node. In this case it is (0+1 = 1).
    - For the second cell, it will display the user name
    - For the third cell, it will display the total price made by the user
        - It uses the sum and key functions to ensure that only the price

values of the current user group under the order element are aggregated.

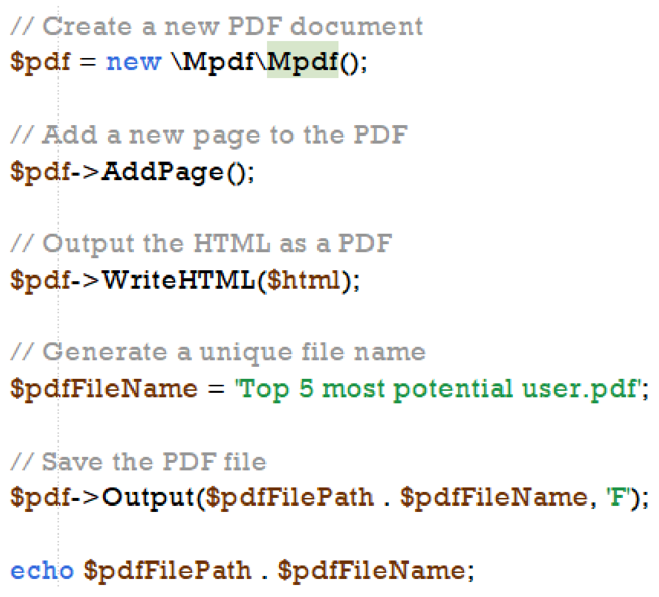
- For the last cell, it will display the total order made by the user
    - In our system, for each order made by the user, their name will be recorded on the particular order
    - Therefore, to count the total order made by the user, we can just use it based on how many times the same user name group appears under the order element.
- After these cells are done setup, means the XLTS file are done, now we will back to the report generation file,



```
//load xsl file
$pdfXsl = new DOMDocument();
$xslFileName = ("potential_user.xsl");
$pdfXsl->load($xslFilePath . $xslFileName);

//create processor and import the stylesheet
$processor = new XSLTProcessor();
$processor->importStylesheet($pdfXsl);

//transform the xml document
$html = $processor->transformToXml($orders);
```

Step 7:

- Now in the report generation file, to process and transform the XML data, first we need to load the created XLTS file and import the stylesheet using the XLTS processor.
- Then, we continue by transforming the loaded XML file using the XLTS processor and store it in a variable.



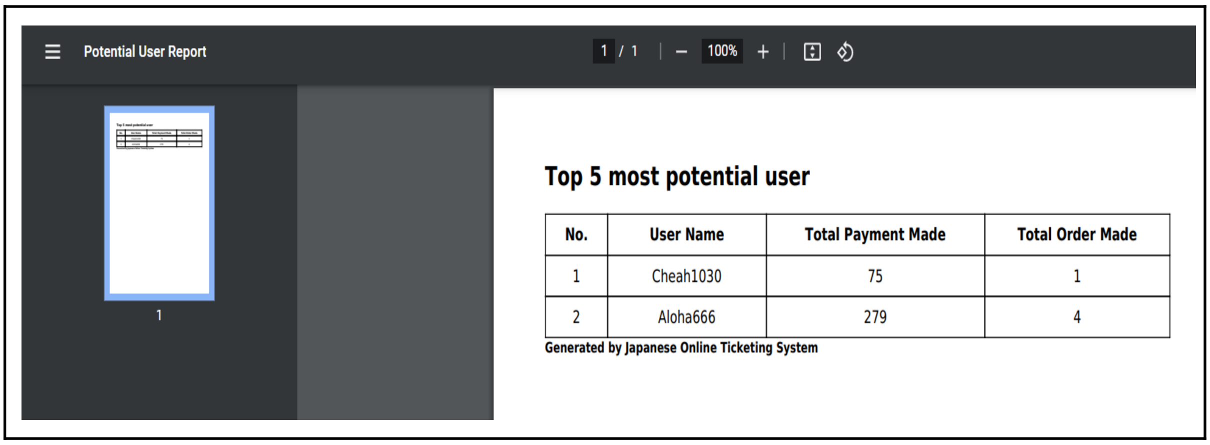
```php
// Create a new PDF document
$pdf = new \Mpdf\Mpdf();

// Add a new page to the PDF
$pdf->AddPage();

// Output the HTML as a PDF
$pdf->WriteHTML($html);

// Generate a unique file name
$pdfFileName = 'Top 5 most potential user.pdf';

// Save the PDF file
$pdf->Output($pdfFilePath . $pdfFileName, 'F');

echo $pdfFilePath . $pdfFileName;
```

Step 8:

- After the data is transformed, we create an instance(which is a new pdf document) of the MPDF class.
    - This pdf instance will be used as a canva to display the transformed pdf table.
- After creating the pdf instance, add a page to the pdf document as by default the page amount is 0.
- Then, we write the transformed data into the pdf document new page.
- Afterwards, we provide a name for the pdf file that will be generated.
- Using the Output function, by providing the path location and filename, save the pdf.
    - The ‘F’ here means the pdf document will be saved
- We then complete this report generation by returning the file location of the newly generated pdf file to the user via an ajax response.
    - Users will be able to view the generated report file in pdf format and choose whether to download it.

6.2 Screenshot

6.2.1 Report



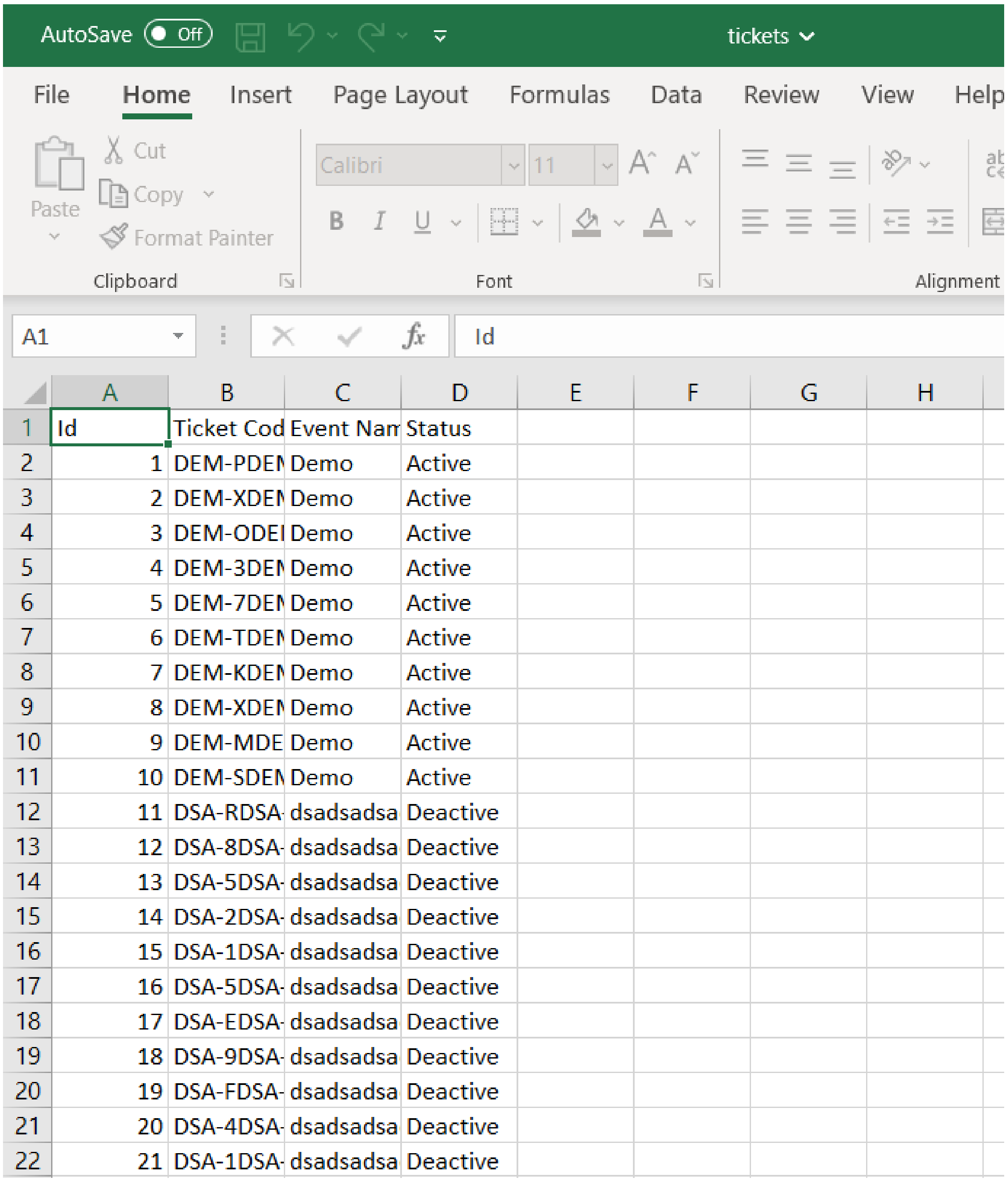
Top 5 most potential user

No.	User Name	Total Payment Made	Total Order Made
1	Cheah1030	75	1
2	Aloha666	279	4

Generated by Japanese Online Ticketing System

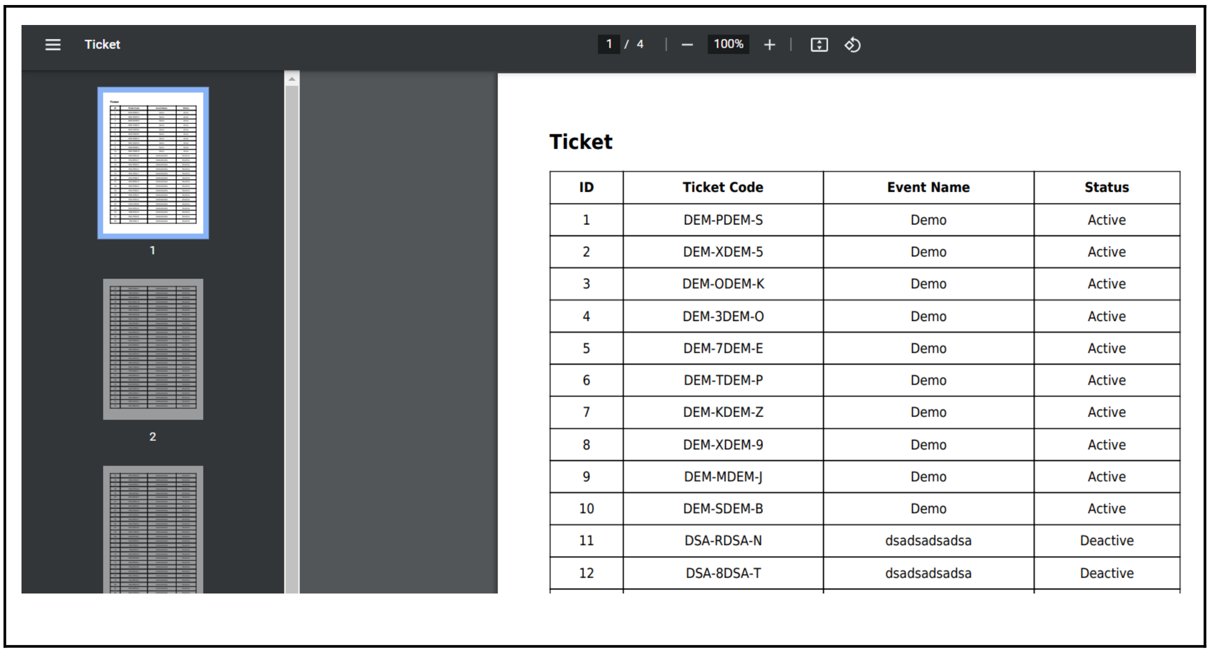
6.2.2 Ticket

CSV File



	A	B	C	D	E	F	G	H
1	Id	Ticket Cod	Event Nam	Status				
2	1	DEM-PDEM	Demo	Active				
3	2	DEM-XDEM	Demo	Active				
4	3	DEM-ODEI	Demo	Active				
5	4	DEM-3DEM	Demo	Active				
6	5	DEM-7DEM	Demo	Active				
7	6	DEM-TDEM	Demo	Active				
8	7	DEM-KDEM	Demo	Active				
9	8	DEM-XDEM	Demo	Active				
10	9	DEM-MDE	Demo	Active				
11	10	DEM-SDEM	Demo	Active				
12	11	DSA-RDSA-	dsadsadsa	Deactive				
13	12	DSA-8DSA-	dsadsadsa	Deactive				
14	13	DSA-5DSA-	dsadsadsa	Deactive				
15	14	DSA-2DSA-	dsadsadsa	Deactive				
16	15	DSA-1DSA-	dsadsadsa	Deactive				
17	16	DSA-5DSA-	dsadsadsa	Deactive				
18	17	DSA-EDSA-	dsadsadsa	Deactive				
19	18	DSA-9DSA-	dsadsadsa	Deactive				
20	19	DSA-FDSA-	dsadsadsa	Deactive				
21	20	DSA-4DSA-	dsadsadsa	Deactive				
22	21	DSA-1DSA-	dsadsadsa	Deactive				

PDF File

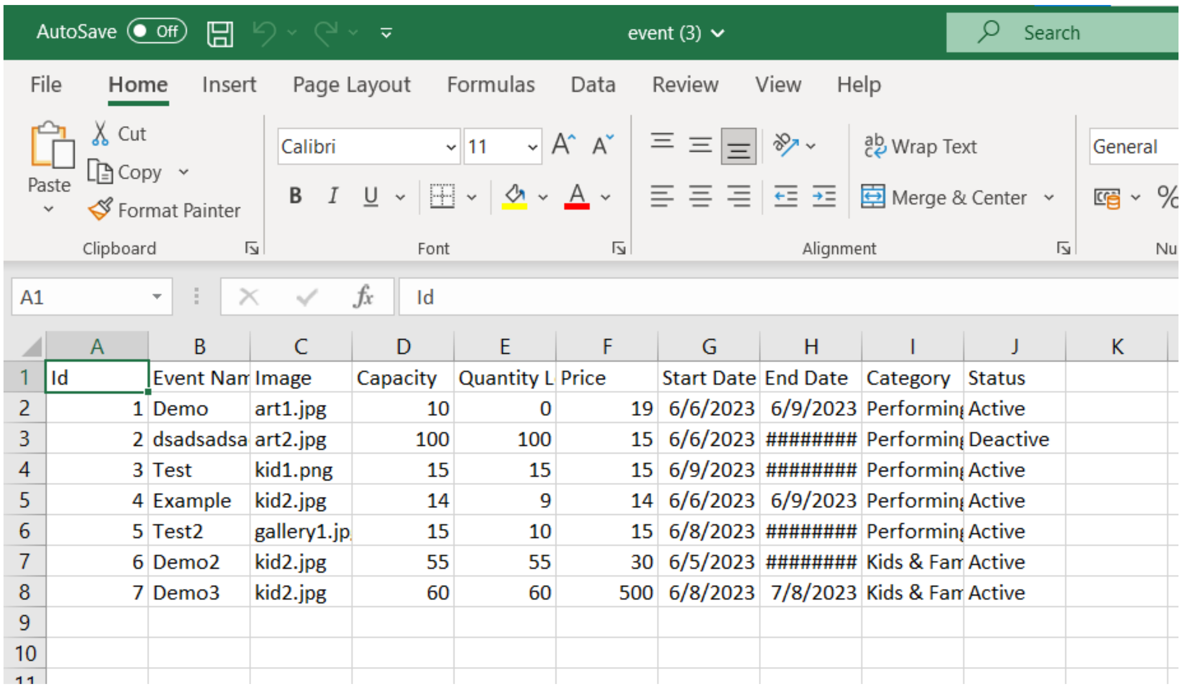


Ticket

ID	Ticket Code	Event Name	Status
1	DEM-PDEM-S	Demo	Active
2	DEM-XDEM-5	Demo	Active
3	DEM-ODEM-K	Demo	Active
4	DEM-3DEM-O	Demo	Active
5	DEM-7DEM-E	Demo	Active
6	DEM-TDEM-P	Demo	Active
7	DEM-KDEM-Z	Demo	Active
8	DEM-XDEM-9	Demo	Active
9	DEM-MDEM-J	Demo	Active
10	DEM-SDEM-B	Demo	Active
11	DSA-RDSA-N	dsadsadsadsa	Deactive
12	DSA-8DSA-T	dsadsadsadsa	Deactive

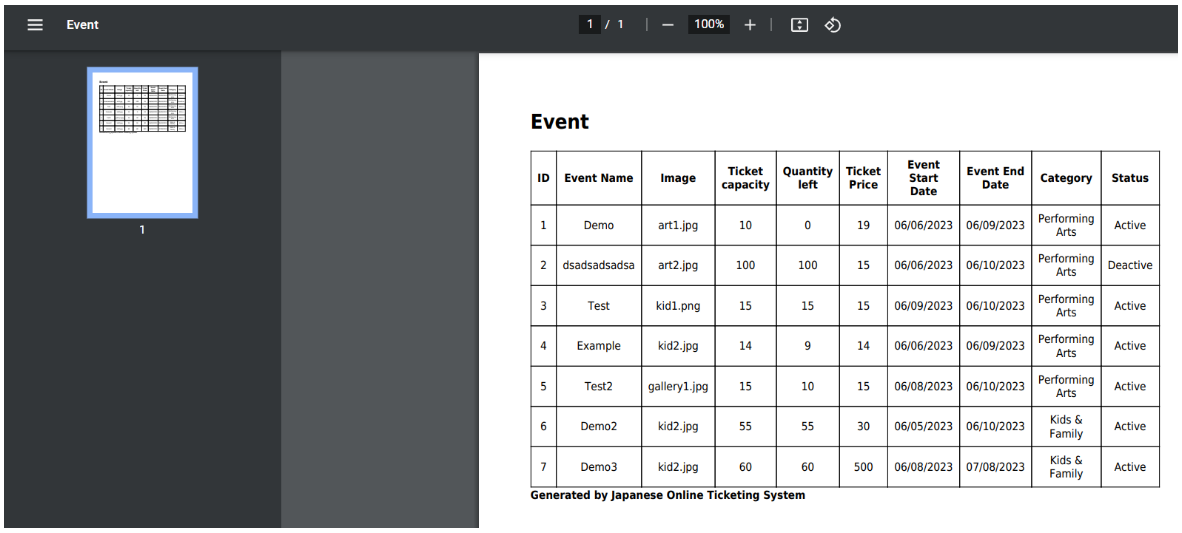
6.2.3 Event

CSV File



	A	B	C	D	E	F	G	H	I	J	K
1	Id	Event Nam	Image	Capacity	Quantity L	Price	Start Date	End Date	Category	Status	
2	1	Demo	art1.jpg	10	0	19	6/6/2023	6/9/2023	Performin	Active	
3	2	dsadsadsa	art2.jpg	100	100	15	6/6/2023	########	Performin	Deactive	
4	3	Test	kid1.png	15	15	15	6/9/2023	########	Performin	Active	
5	4	Example	kid2.jpg	14	9	14	6/6/2023	6/9/2023	Performin	Active	
6	5	Test2	gallery1.jp	15	10	15	6/8/2023	########	Performin	Active	
7	6	Demo2	kid2.jpg	55	55	30	6/5/2023	########	Kids & Fam	Active	
8	7	Demo3	kid2.jpg	60	60	500	6/8/2023	7/8/2023	Kids & Fam	Active	
9											
10											

PDF File



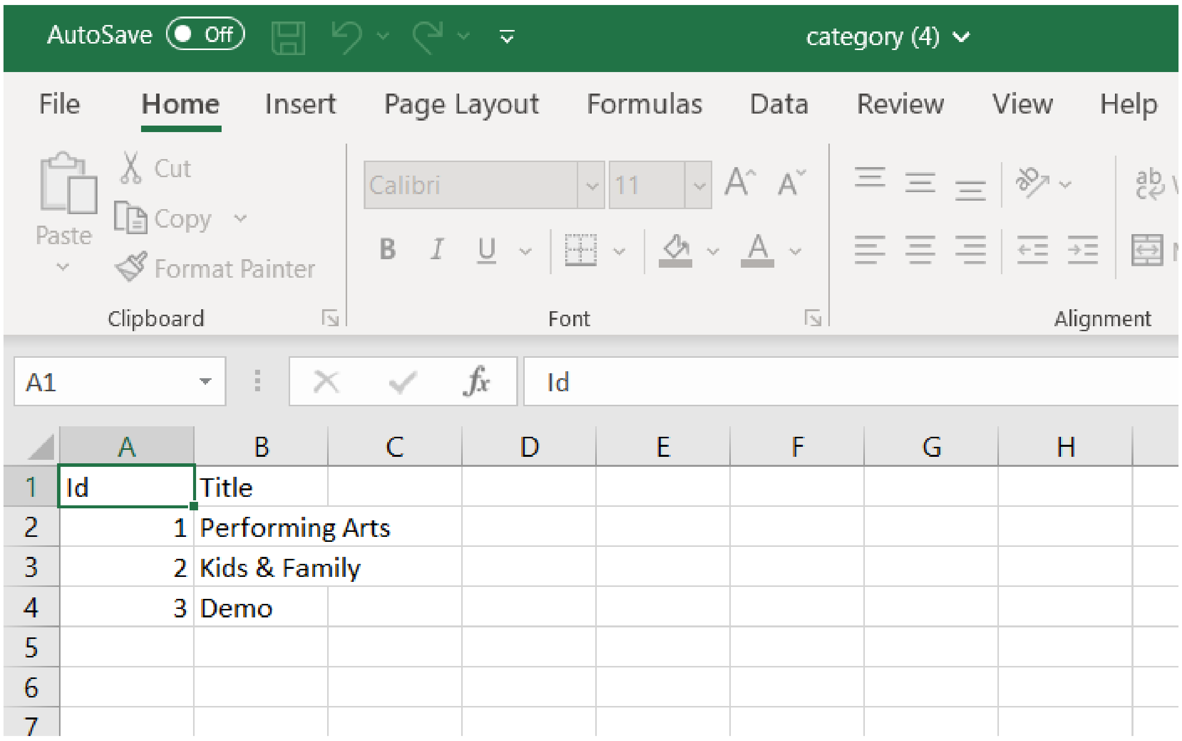
Event

ID	Event Name	Image	Ticket capacity	Quantity left	Ticket Price	Event Start Date	Event End Date	Category	Status
1	Demo	art1.jpg	10	0	19	06/06/2023	06/09/2023	Performing Arts	Active
2	dsadsadsadsa	art2.jpg	100	100	15	06/06/2023	06/10/2023	Performing Arts	Deactive
3	Test	kid1.png	15	15	15	06/09/2023	06/10/2023	Performing Arts	Active
4	Example	kid2.jpg	14	9	14	06/06/2023	06/09/2023	Performing Arts	Active
5	Test2	gallery1.jpg	15	10	15	06/08/2023	06/10/2023	Performing Arts	Active
6	Demo2	kid2.jpg	55	55	30	06/05/2023	06/10/2023	Kids & Family	Active
7	Demo3	kid2.jpg	60	60	500	06/08/2023	07/08/2023	Kids & Family	Active

Generated by Japanese Online Ticketing System

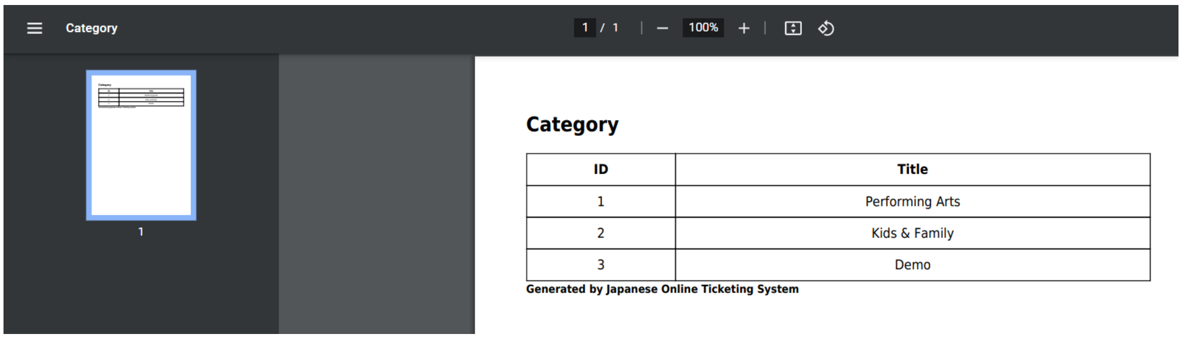
6.2.4 Category

CSV File



	A	B
1	Id	Title
2	1	Performing Arts
3	2	Kids & Family
4	3	Demo

PDF File



Category

ID	Title
1	Performing Arts
2	Kids & Family
3	Demo

Generated by Japanese Online Ticketing System

7. Web Services

For my part, I have implemented Restful web service technology to the Category module. To implement the RESTFUL API, I took some steps:

1. I have defined the API endpoints which is “category”, so my url will be in the format of “/category”. This means that users have to access the related information through this url instead of “/users” or “/status”.
2. After this, I choose to use the GET method to perform specific actions on the available resources. For example, “/users/View/update.php?title=”.
3. Since I'm working on the php language, I decided to use JSON format to fetch my data from page to page.
4. Moreover, I used 4 HTTP status codes which are 200, 400, 404 and 500. 200 used to represent successful user requests, 400 represented invalid user action, 404 is used for data not found while 500 is for internal server errors.
5. Then, I have implemented the necessary CRUD operations such as “DisplayAll”, “DisplaySingle”, “Insert”, “Update” and “Delete” functions for users to manage the category resources.

So these are the overall steps that I included in order to implement a successful RESTFUL API.

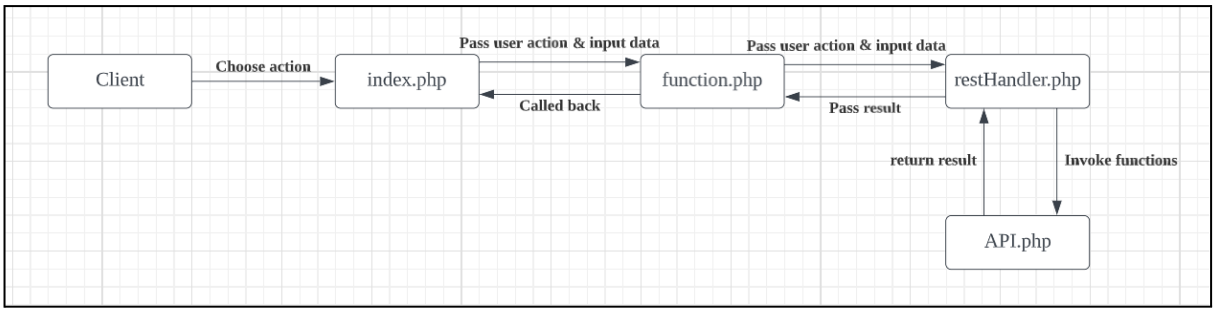
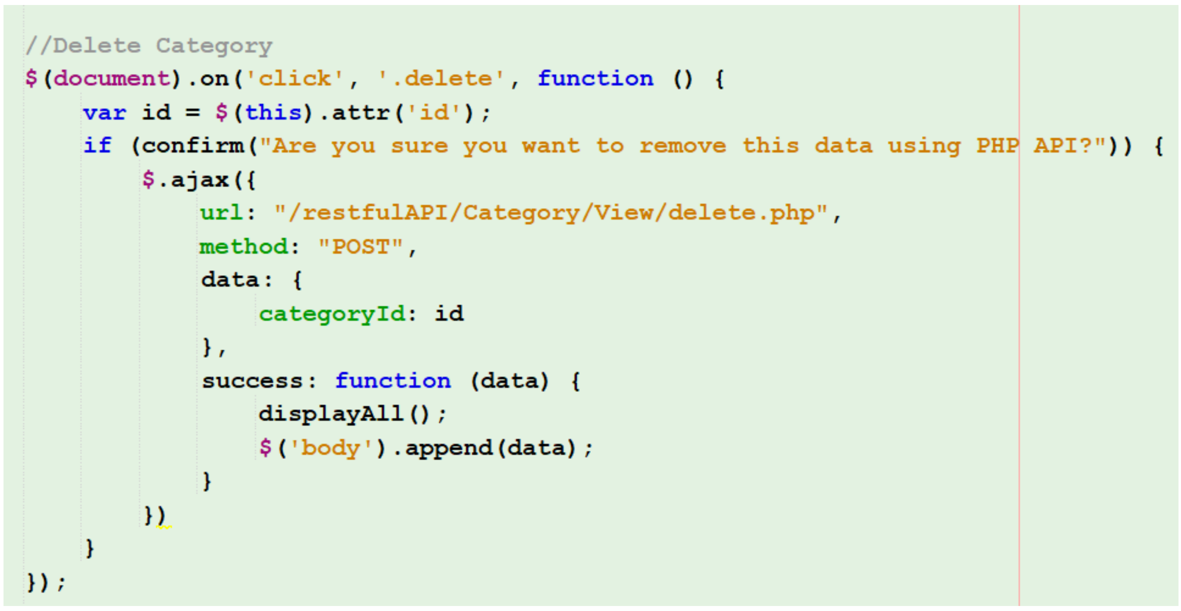


Figure 2.0: Overview of RESTful APi for Category module

Code Snippet:

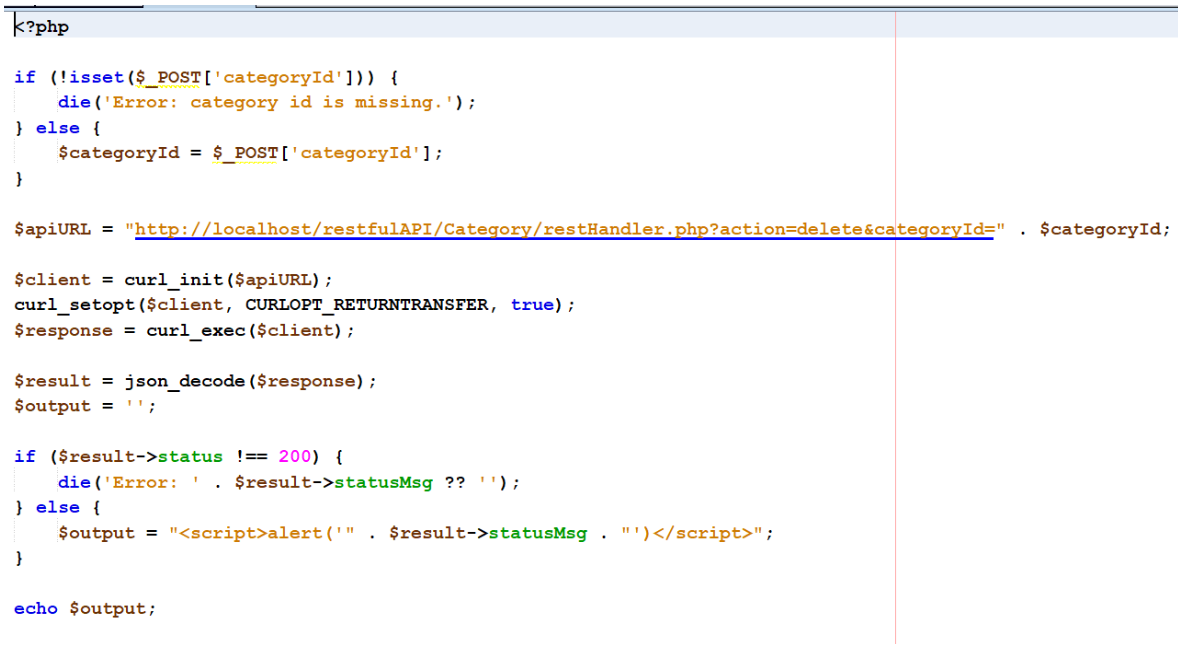


```
//Delete Category
$(document).on('click', '.delete', function () {
    var id = $(this).attr('id');
    if (confirm("Are you sure you want to remove this data using PHP API?")) {
        $.ajax({
            url: "/restfulAPI/Category/View/delete.php",
            method: "POST",
            data: {
                categoryId: id
            },
            success: function (data) {
                displayAll();
                $('body').append(data);
            }
        })
    }
});
```

Explanation:

In this example, I will use the “delete” function to explain the flow of how the RESTFUL API works in my category module. From figure 2.0, clients will only interact with index.php so this file will be in charge of passing the user requests to other files. Other than this, it will also be the main page which used to display the result in table format. So, when the user clicks the delete button, the application will show a confirmation message to the user. Once the user confirms to delete, the application will forward to the delete.php (function.php) and attach it with the category id.

Code Snippet:

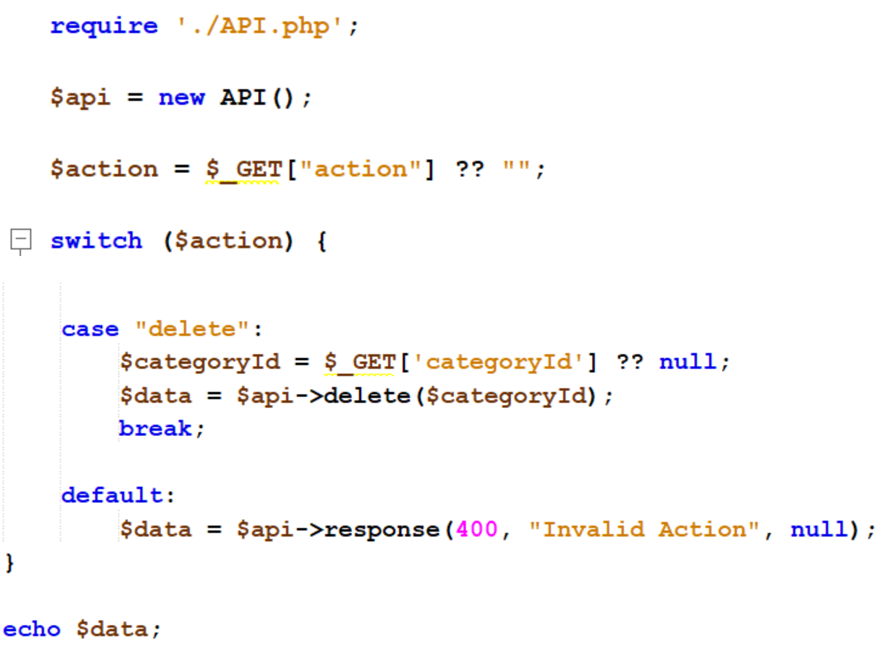


```
<?php

if (!isset($_POST['categoryId'])) {
    die('Error: category id is missing.');
} else {
    $categoryId = $_POST['categoryId'];
}

$apiURL = "http://localhost/restfulAPI/Category/restHandler.php?action=delete&categoryId=" . $categoryId;

$client = curl_init($apiURL);
curl_setopt($client, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($client);

$result = json_decode($response);
$output = '';

if ($result->status !== 200) {
    die('Error: ' . $result->statusMsg ?? '');
} else {
    $output = "<script>alert('" . $result->statusMsg . "')</script>";
}

echo $output;
```

Explanation:

In this delete.php, it will first check whether any categoryId received, otherwise it will return to the user an error message and stop the subsequent operation. There will be a variable named “$apiURL” which contains the destination of restHander.php and its action associated with the categoryId by using the GET method. Then, this url will be executed on the “curl_exec()” function.

Code Snippet:

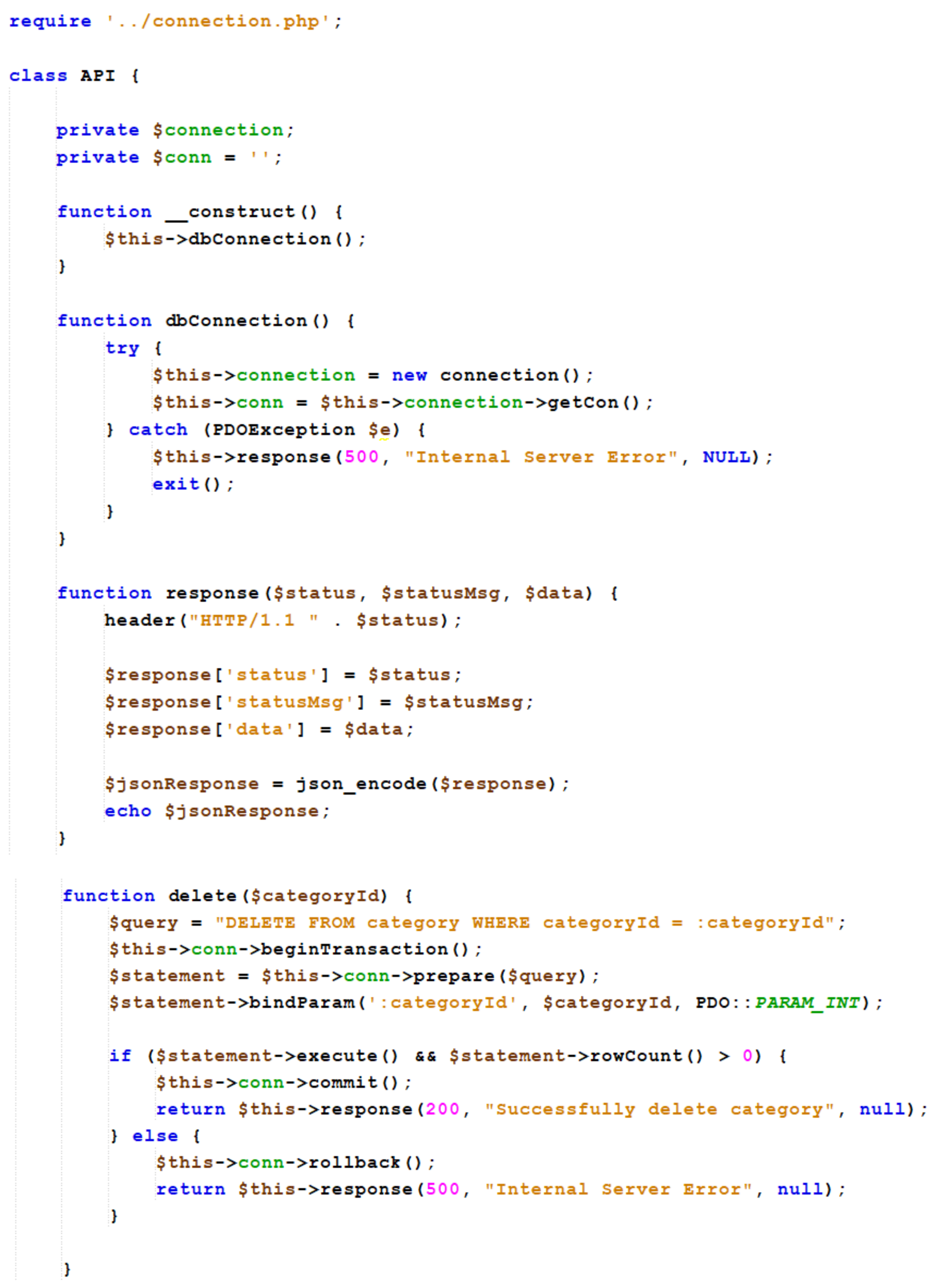


```
    require './API.php';

    $api = new API();

    $action = $_GET["action"] ?? "";

    switch ($action) {

        case "delete":
            $categoryId = $_GET['categoryId'] ?? null;
            $data = $api->delete($categoryId);
            break;

        default:
            $data = $api->response(400, "Invalid Action", null);
}

echo $data;
```

Explanation:

This is the screenshot of restHandler.php which is used to handle different user actions. In this case, the variable “$action” will get the “delete” string and it will execute the code which is “$api->delete()”. This means that it will invoke the API delete method/operation and save the return result in “$data”. Also, if there is invalid user action detected, it will also have the 400 status code. But at this moment, the result and data is save in the “$data” variable, nothing will be printed or displayed in this file.

Code Snippet:



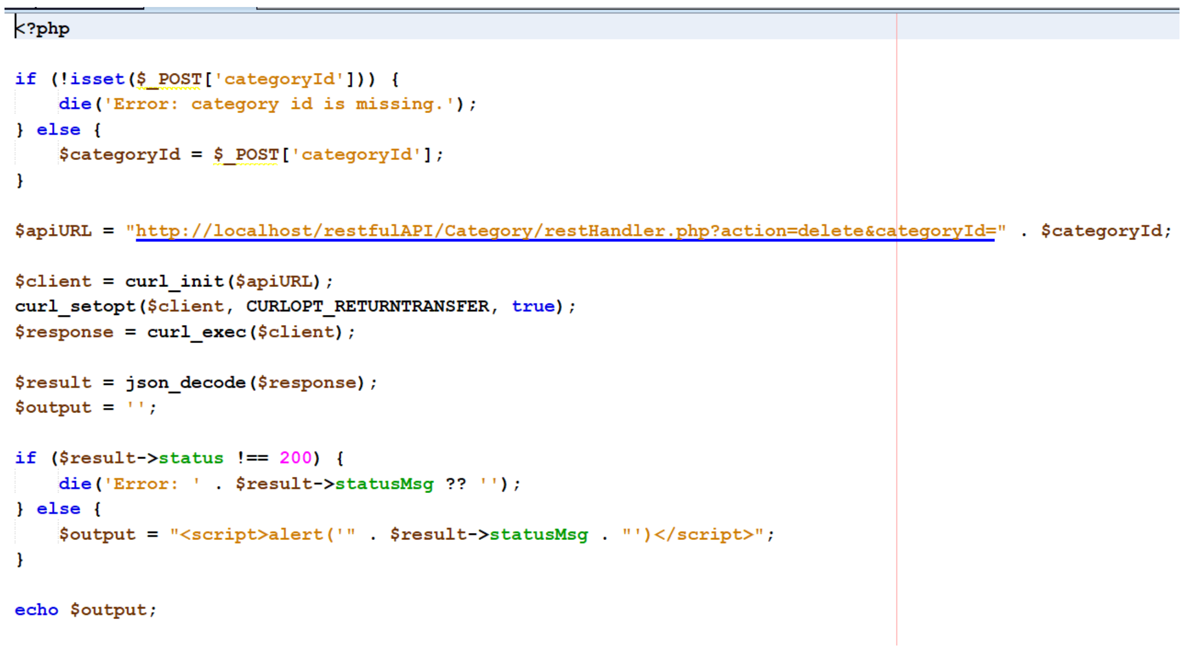
```php
require '../connection.php';

class API {

    private $connection;
    private $conn = '';

    function __construct() {
        $this->dbConnection();
    }

    function dbConnection() {
        try {
            $this->connection = new connection();
            $this->conn = $this->connection->getCon();
        } catch (PDOException $e) {
            $this->response(500, "Internal Server Error", NULL);
            exit();
        }
    }

    function response($status, $statusMsg, $data) {
        header("HTTP/1.1 " . $status);

        $response['status'] = $status;
        $response['statusMsg'] = $statusMsg;
        $response['data'] = $data;

        $jsonResponse = json_encode($response);
        echo $jsonResponse;
    }

     function delete($categoryId) {
         $query = "DELETE FROM category WHERE categoryId = :categoryId";
         $this->conn->beginTransaction();
         $statement = $this->conn->prepare($query);
         $statement->bindParam(':categoryId', $categoryId, PDO::PARAM_INT);

         if ($statement->execute() && $statement->rowCount() > 0) {
             $this->conn->commit();
             return $this->response(200, "Successfully delete category", null);
         } else {
             $this->conn->rollback();
             return $this->response(500, "Internal Server Error", null);
         }

     }
```

Explanation:

This is the code in API.php which includes database connection, CRUD operation and response method. From the above scenario, when the restHandler.php invokes the “delete” function, it will execute the sql query first. Then the corresponding status code, message and data will be returned based on the situation. The result will be stored in

“$response” in JSON format. Afterwards, the result will be received by the “$data” in restHandler.php.

Code Snippet:

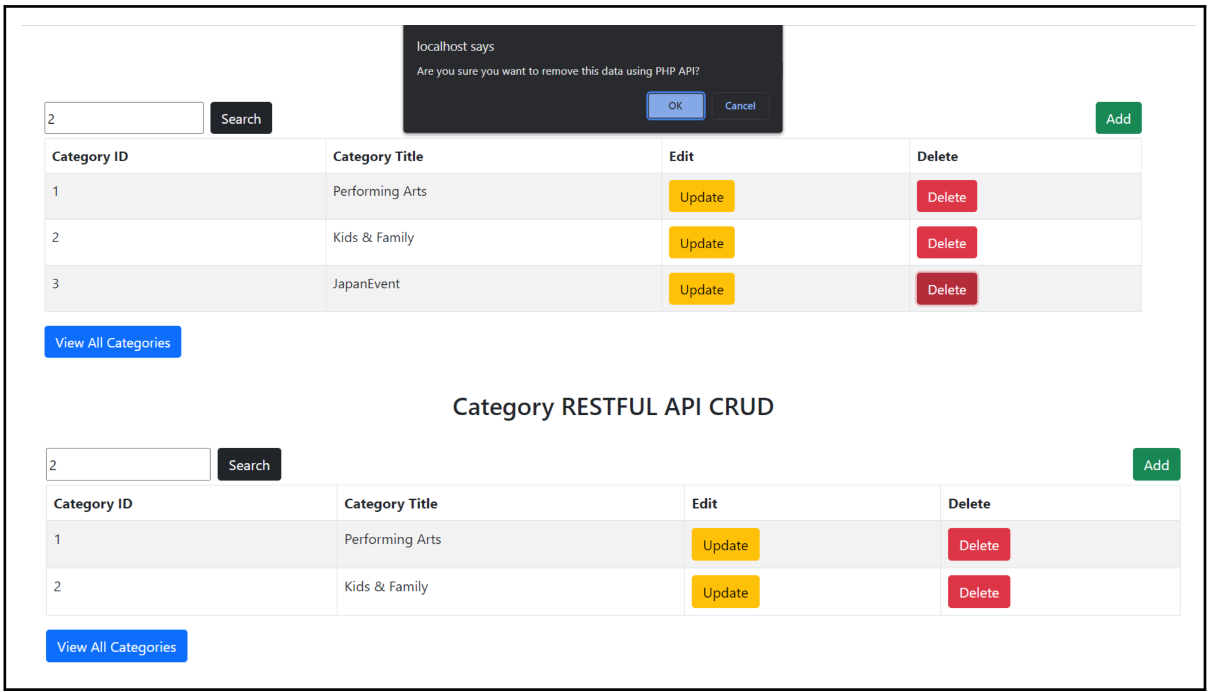


```php
<?php

if (!isset($_POST['categoryId'])) {
    die('Error: category id is missing.');
} else {
    $categoryId = $_POST['categoryId'];
}

$apiURL = "http://localhost/restfulAPI/Category/restHandler.php?action=delete&categoryId=" . $categoryId;

$client = curl_init($apiURL);
curl_setopt($client, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($client);

$result = json_decode($response);
$output = '';

if ($result->status !== 200) {
    die('Error: ' . $result->statusMsg ?? '');
} else {
    $output = "<script>alert('" . $result->statusMsg . "')</script>";
}

echo $output;
```

Explanation:

Then, the “$data” in restHandler.php will finally be passed back to the delete.php and be decoded. So, if it is a valid status code (200), then the result and status message will be returned to the user. Lastly, the index.php will call back this function to get the result and display it in the empty table body.

Snapshot of the delete output



localhost says

Are you sure you want to remove this data using PHP API?

OK Cancel

2 Search Add

Category ID	Category Title	Edit	Delete
1	Performing Arts	Update	Delete
2	Kids & Family	Update	Delete
3	JapanEvent	Update	Delete

View All Categories

Category RESTFUL API CRUD

2 Search Add

Category ID	Category Title	Edit	Delete
1	Performing Arts	Update	Delete
2	Kids & Family	Update	Delete

View All Categories

**Please note that this is only part of the whole code*

8. Conclusion

In this task, I applied a variety of techniques, some of which made my job easier and allowed me to achieve my goal quickly. Firstly, is the design pattern that I have applied in this assignment, which is the proxy pattern. One of the things I like about the proxy pattern is that it reduces the system workload. Because it will cache loaded objects (images in this case). Once I request the same object, the modal displays the object again from the cache instead of loading the object again.

Next, I'll talk about the web services technology I'm using in this assignment, the Restful API. Restful API helped me a lot in applying the web service, It gives me a standardized interface to get things done quickly. It also provides me with several HTTP methods that allow me to quickly achieve my goal, such as HTTP GET and POST methods.

The last technology I'm happy to use is the XLTS language for generating reports. It allows me to design templates for flexible transformation of XML data. With the help of this language, I managed to turn XML data into a nice PDF table format.

BAIT3173 Integrative Programming - Assignment Rubrics

Name: Lee Ting Le

Programme/Group: RSD2G2

Japanese Online Ticketing System

Event, ticket and category

No.	Aspect	Developing 0 - 4 marks	Approaching 5-7 marks	Ideal 8-10 marks	Remarks	Score
1	**PHP Web Application**	Overview & module description reflects a beginner level in terms of personal skills and character.	Overview & module description reflects an intermediate level in terms of personal skills and character.	Overview & module description reflects an ideal level in terms of personal skills and character.		
		Implementation of the system demonstrates a beginner level in this aspect/technology.	Implementation of the system demonstrates a moderate level of competence in this aspect/technology.	Implementation of the system demonstrates a high level of competence in this aspect/technology.		
2	**Design Patterns**	Description reflects a beginner level in terms of personal skills and character.	Description reflects an intermediate level in terms of personal skills and character.	Description reflects an ideal level in terms of personal skills and character.		
		Implementation of the system demonstrates a beginner level in this aspect/technology.	Implementation of the system demonstrates a moderate level of competence in this aspect/technology.	Implementation of the system demonstrates a high level of competence in this aspect/technology.		
3	**Secure Coding Practices**	Description reflects a beginner level in terms of personal skills and character.	Description reflects an intermediate level in terms of personal skills and character.	Description reflects an ideal level in terms of personal skills and character.		
		Implementation of the system demonstrates a beginner level in this aspect/technology.	Implementation of the system demonstrates a moderate level of competence in this aspect/technology.	Implementation of the system demonstrates a high level of competence in this aspect/technology.		

4	**XML, XSLT & XPath**	Description reflects a beginner level in terms of personal skills and character.	Description reflects an intermediate level in terms of personal skills and character.	Description reflects an ideal level in terms of personal skills and character.		
		Implementation of the system demonstrates a beginner level in this aspect/technology.	Implementation of the system demonstrates a moderate level of competence in this aspect/technology.	Implementation of the system demonstrates a high level of competence in this aspect/technology.		
5	**Web Services**	Description reflects a beginner level in terms of personal skills and character.	Description reflects an intermediate level in terms of personal skills and character.	Description reflects an ideal level in terms of personal skills and character.		
		Implementation of the system demonstrates a beginner level in this aspect/technology.	Implementation of the system demonstrates a moderate level of competence in this aspect/technology.	Implementation of the system demonstrates a high level of competence in this aspect/technology.		

Note: **Total / 100**

Personal skills: independent learning, intellectual and self-development; confidence, self-control, social skills, proper etiquette and commitment to professionalism. Includes **character values** such as honesty, punctuality, time management as well as keeping to and maintaining deadlines.
Practical skills: competence and professional practice in planning (including study & research), design and implementation of technology.