5.9 Implementation Sample Codes:

Login Functionality:

```
Login Sample Code
<?php
require 'assets/conn.php';
if(isset($_POST['submit'])){
$id=$_POST['id'];
$password=$_POST['password'];
$select="SELECT * FROM employee WHERE id ='$id' AND password='$password' AND status='1'";
$query=mysqli_query($conn,$select) or die(mysqli_error($conn));
$select_admin="SELECT * FROM admin WHERE id ='$id' AND password='$password' AND status='1'";
$admin_query=mysqli_query($conn,$select_admin) or die(mysqli_error($conn));
if(mysqli_num_rows($query)>0){
    $_SESSION['emp_id']=$_POST['id'];
    header('location:employees-dashboard.php');
}elseif(mysqli_num_rows($admin_query)>0){
    $_SESSION['admin_id']=$_POST['id'];
    header('location:employees.php');
}else{
    echo"<script>alert('ID OR PASSWORD IS INCORRECT')</script>";
}
}

...
?>
```

Leave Functionality:

```
Leave Sample Code:
<?php
...
                                                    <tbody>';
$select="SELECT * FROM le";
$query=mysqli_query($conn,$select) or die(mysqli_error($conn));
if(mysqli_num_rows($query)>0){
    while ($row=mysqli_fetch_assoc($query)) {
        echo'<tr>
            <td>'.$row['date'].'</td>
            <td>'.$row['id'].'</td>
            <td>'.nl2br($row['reason']).'</td>
        </tr>';
    }}

...
?>
```

Database Connection and Activity Code:

```
Connection Path: ./assets/conn.php
<?php
session_start();
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "empops";
$conn = mysqli_connect($servername, $username, $password, $dbname);
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
?>

Code for Database Connection and Activity
<?php
require 'assets/conn.php';
if(isset($_POST['inactive'])){
  $eid=$_POST['id'];
  $update="UPDATE employee SET status = '2' WHERE id='$eid'";
  mysqli_query($conn,$update) or die(mysqli_error($conn));
  if(isset($_SESSION['emp_id'])){
  header('location:employees-list.php');
}else{
    header('location:emp-list.php');
}
}
if(isset($_POST['active'])){
  $eid=$_POST['id'];
  $update="UPDATE employee SET status = '1' WHERE id='$eid'";
  mysqli_query($conn,$update) or die(mysqli_error($conn));
    if(isset($_SESSION['emp_id'])){
  header('location:employees-list.php');
}else{
  header('location:emp-list.php');
}

...
?>
```

Database Connection and Activity Code:

```
Connection Path: ./assets/conn.php
<?php
session_start();
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "empops";
$conn = mysqli_connect($servername, $username, $password, $dbname);
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
?>

Code for Database Connection and Activity
<?php
require 'assets/conn.php';
if(isset($_POST['inactive'])){
  $eid=$_POST['id'];
  $update="UPDATE employee SET status = '2' WHERE id='$eid'";
  mysqli_query($conn,$update) or die(mysqli_error($conn));
  if(isset($_SESSION['emp_id'])){
  header('location:employees-list.php');
}else{
    header('location:emp-list.php');
}
}
if(isset($_POST['active'])){
  $eid=$_POST['id'];
  $update="UPDATE employee SET status = '1' WHERE id='$eid'";
  mysqli_query($conn,$update) or die(mysqli_error($conn));
    if(isset($_SESSION['emp_id'])){
  header('location:employees-list.php');
}else{
  header('location:emp-list.php');
}

...
?>
```

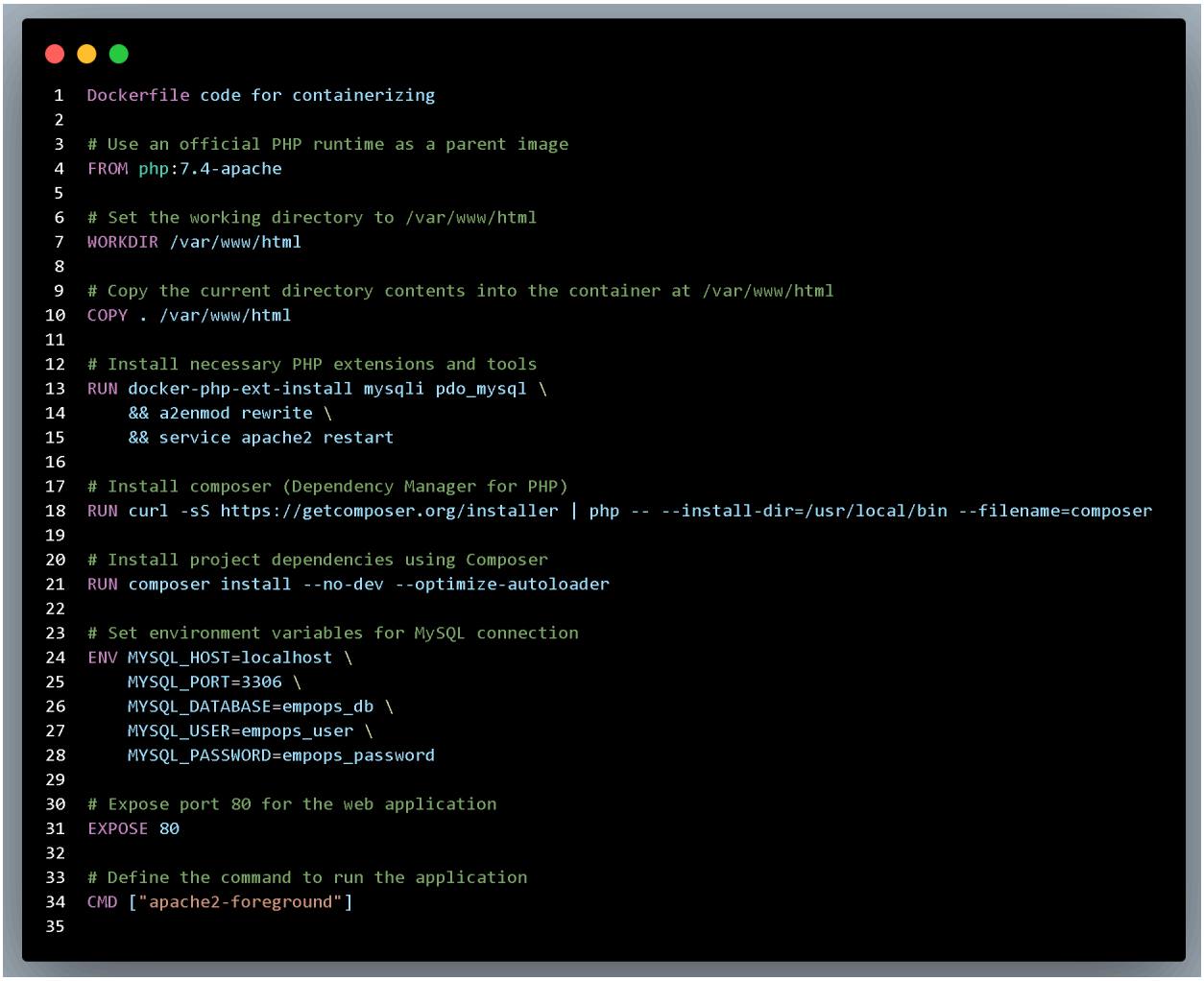
Database Query and Operations like Updating, Inserting and Deleting Datas Sample Code:

```
Some Database Operations to make Query, Update and Insert Datas
<?php
if(isset($_POST['add_company'])){
  $name=$_POST['company'];
  $r_n=$_POST['r_n'];
  $i_date=$_POST['i_date'];
  $vat=$_POST['vat'];
  $address=$_POST['address'];
  $phone=$_POST['phone'];
  $email=$_POST['email'];
  $web=$_POST['web'];
$update="UPDATE company SET name='$name' , r_n = '$r_n' ,
                 i_date='$i_date' , vat='$vat' , address='$address'
                 , phone='$phone' , email='$email' , web='$web'";
mysqli_query($conn,$update) or die(mysqli_error($conn));
header('location:company-admin.php');
}
if(isset($_POST['upload'])){
  $name=$_POST['name'];
  $size=$_POST['size'];
  $date=$_POST['date'];
  $url=$_POST['url'];
    $insert="INSERT INTO document(name,date,size,link) VALUES('$name','$size','$date','$url')";
    mysqli_query($conn,$insert) or die(mysqli_error($conn));
  header('location:company-admin.php');
}
if(isset($_POST['delete'])){
  $id=$_POST['id'];
  $delete="DELETE FROM document WHERE id ='$id'";
  mysqli_query($conn,$delete) or die(mysqli_error($conn));
  header('location:company-admin.php');
}
if(isset($_POST['apply'])){
  $id=$_POST['id'];
  $date=$_POST['date'];
  $reason=nl2br($_POST['reason']);
  $delete="DELETE FROM employee WHERE id = '$id'";
  mysqli_query($conn,$delete) or die(mysqli_error($conn));
  $select="SELECT * FROM employee WHERE id = '$id'";
  $sec=mysqli_query($conn,$select) or die(mysqli_error($conn));
  if(mysqli_num_rows($sec)>0){
    $insert="INSERT INTO le(id,date,reason) VALUES('$id','$date','$reason')";
  mysqli_query($conn,$insert);
}

...
?>
```

Composer json sample code:

```
Composer file code
{
    "name": "devops-project/empops",
    "description": "Employee Management System",
    "type": "project",
    "license": "MIT",
    "authors": [
        {
            "name": "Gemechis",
            "email": "gemechis@example.com"
        }
    ],
    "require": {
        "php": "^7.4",
        "slim/slim": "^4.8",
        "slim/psr7": "^1.5",
        "illuminate/database": "^8.0",
        "vlucas/phpdotenv": "^5.4"
    },
    "autoload": {
        "psr-4": {
            "YourNamespace\\": "src/"
        }
    },
    "scripts": {
        "post-install-cmd": [
            "Illuminate\\Database\\Events\\PostgresNotificationServiceProvider::boot"
        ]
    },
    "config": {
        "optimize-autoloader": true,
        "preferred-install": "dist",
        "sort-packages": true
    }
}

```

Dockerfile sample Code for Containerization:



```
Dockerfile code for containerizing

# Use an official PHP runtime as a parent image
FROM php:7.4-apache

# Set the working directory to /var/www/html
WORKDIR /var/www/html

# Copy the current directory contents into the container at /var/www/html
COPY . /var/www/html

# Install necessary PHP extensions and tools
RUN docker-php-ext-install mysqli pdo_mysql \
    && a2enmod rewrite \
    && service apache2 restart

# Install composer (Dependency Manager for PHP)
RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer

# Install project dependencies using Composer
RUN composer install --no-dev --optimize-autoloader

# Set environment variables for MySQL connection
ENV MYSQL_HOST=localhost \
    MYSQL_PORT=3306 \
    MYSQL_DATABASE=empops_db \
    MYSQL_USER=empops_user \
    MYSQL_PASSWORD=empops_password

# Expose port 80 for the web application
EXPOSE 80

# Define the command to run the application
CMD ["apache2-foreground"]

```

Jenkinfile for Continuous Integration Pipeline Script :

```
Jenkinfile for Continuous Integration Pipeline Script

pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                // Checkout the source code from the version control system
                git 'https://github.com/DevOps-Project/empops.git'
            }
        }

        stage('Install Dependencies') {
            steps {
                // Run Composer to install PHP dependencies
                sh 'composer install --no-dev --optimize-autoloader'
            }
        }

        stage('Run Tests') {
            steps {
                // Run PHPUnit tests
                sh 'vendor/bin/phpunit'
            }
        }

        stage('Build and Deploy') {
            steps {
                // Build and deploy the application (adjust as per your deployment process)
                // For example, you might copy files to the web server directory
                sh 'cp -r * /var/www/html'
            }
        }
    }

    post {
        success {
            // Trigger additional actions on successful build
            echo 'Build successful! Deploying...'
        }
        failure {
            // Notify or take actions on build failure
            echo 'Build failed! Notify the team...'
        }
    }
}

```

CHAPTER SIX:
CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

In conclusion, the development and implementation of the EmpOps Employee Management System marks a significant milestone in transforming traditional employee management practices. Throughout this project, key observations and outcomes have shaped our understanding of the system's impact and effectiveness.
The EmpOps system addresses critical challenges faced by organizations, including manual data handling, communication inefficiencies, and limited access to vital information. By leveraging technologies such as HTML, CSS, JS, Bootstrap, PHP, MySQL, and DevOps tools like Jenkins, Git, Docker, and GitHub, we've created a robust and user-friendly platform.

The system introduces features such as streamlined authentication, role-based access control, comprehensive dashboards, employee profiles, leave management, and integration with DevOps practices. The emphasis on continuous integration and deployment ensures a seamless and efficient development lifecycle.
The journey of EmpOps has not only resulted in a functional employee management system but also enriched our understanding of collaborative development, DevOps integration, and the importance of user-centric design. The feedback received from various testing phases and user interactions has been instrumental in refining the system's functionality and user experience.

While EmpOps represents a significant achievement, it also opens avenues for future enhancements. As technology evolves and user needs expand, continuous development and refinement will be crucial to maintaining EmpOps as a cutting-edge solution for employee management.
In the next section, we present recommendations based on our experiences and observations during the development of EmpOps. These recommendations aim to guide future iterations and improvements, ensuring the sustained success of the system.