Cloud Application Development Project

Host a Cloud Application

Daniel Jameson T00158237

2025

# Introduction

Note: This write-up document is considered to be an “Academic Research” Document, treat it accordingly.

All code used in this project can be found at the following Github Repository:

<https://github.com/DogPope/CloudPhpApp.git>

\*NOTE: Update this section before submission. There is also cloud related things need to be added here that are not decided yet.

Cloud Computing is the process of moving the hosting of applications to off-site servers that allow for lower overheads, such as in the areas of security and physical space.

Cloud Computing offers potential customers many advantages that traditional servers do not. Cloud Computing allows potential customers to host their services in data centres and not locally. This allows for the reduction in costs by not requiring hardware to store data, delegating that task to others and making the organisation more flexible as a result.

Cloud computing has also introduced risks associated with hosting your application on servers in terms of reliability, and costs.

# Introduction: Chosen Application

The application for this assignment is a simple CRUD application written in the Hypertext Preprocessor language (PHP). This application focuses on selling games to customers, who can sign up to an account, log in to the service and use their credentials to purchase games from a database.

The database language used was MySql, in this case the version that comes with Xampp, an Apache service. The service implemented within Xampp is called MariaDB, a fork of MySql that was taken before it’s acquisition by Oracle.

The business logic in this application is implemented in PHP. PHP is a weakly typed language, designed to work in Web Browsers. It shares this space in the market with the Javascript programming language, and is considered overshadowed by the Javascript programming language.

The application was refactored over the course of a week to update the file structure to reflect more modern structures and introduce a more sane layout that is more intuitive to the end user. The application logic was also updated to ensure that all functionality was accounted for and would provide a less frictional experience going forward.

For ergonomical purposes, the versions of the application will have different colour schemes.

The application starting point is very brightly coloured, using light green for the Navigation and light yellow for the main page content. Over time, this colour scheme will change to a darker scheme, as it is easier on the eyes and simpler in its layout, requiring less fields to be entered and simplifying the data transmission for ease of use.

# Chosen Cloud Provider

What does the chosen cloud provider need to accomplish to be a suitable platform for a PHP application? The standard with PHP apps is deploying on a local server using Apaches’ Xampp service. This allows an application to communicate over a standard port number, run on a basic web service. In this case, the App needs to communicate from the default port number at port 80.

The application has some aspects that would benefit would a cloud deployment. The application could benefit from services such as Elastic Beanstalk and others.

# Migrating the Application to a Cloud Environment

The application can be copied in its’ current state to Amazon Web Services. The application could be easily deployed using the “Elastic Beanstalk” service on AWS. This process will involve the setting up of many distinct services within the AWS environment. The initial setup for this could include setting up a Virtual Private Cloud service to manage traffic and communication between different parts of the application.

## - Migrating the Database

Migrating a MariaDB database is a complicated process, involving setting up the underlying infrastructure first. The networking solutions need to be made in such a way that will not compromise the security of the application. Creating a Database Subnet has to be done first, as the necessary availability zones must be created and be possible to access. This means that the database can be deployed across multiple Availability Zones(AZ) and never truly goes down for the end user. In this case, the EC2 instance was not connected, as a manual connection would be more appropriate for the purpose of gradually migrating parts of the application.

## Database Migration Service

Migrating the database involves adding

# Strangulation Design Pattern

While this application is good enough for a second year Server-Side Development project, it should be considered to be outdated and insecure. The application uses several unsafe and suboptimal methods in its execution. Some security concerns were taken into account when the application was made, such as the use of Prepared Statements in the code for the purpose of preventing an SQL injection attack. An example of this can be seen below: A computer code on a black background

AI-generated content may be incorrect.

These statements have a lot of information to unpack, and there are good and bad attributes to any given SQL statement in the system. The use of Prepared Data Objects for queries is a good method of preventing SQL injection, but this setup exposes valuable secrets and information that should not be exposed to any actors not actively working on the application. The final two arguments in the PDO statement contain two such secrets, the ‘root’ username and the ‘’ (blank) password for accessing the database.

The Strangulation Design Pattern could be considered a good solution to this problem. The Strangulation Design Pattern takes an existing application and aims to refactor it, over time, into a new codebase(CITATION).

Companies have used this pattern in the past to great effect, improving old codebases incrementally, gradually morphing those systems into something more modern and usable. This approach allowed these companies to refactor the entire codebase over time, providing the given application with new approaches to building, allowing the use of newer design patterns that better reflect the modern development landscape, such as going from a monolithic architecture to a microservices based architecture (Hammant, 2013).

The question, for this report, is how can this refactoring be implemented within the given AWS services. Luckily, Amazon maintains documentation on how this transition can be accomplished on their own services (Amazon, 2025). Following this, the application should be transferrable to a more modern, fully functional Web Application.

# Refactoring the Application on AWS

## Applying a Framework

The most obvious starting point for such an application would be the migration to the “Laravel” framework for PHP. Following that, the rest of the application needs cleaning and tidying, such as the aforementioned security vulnerabilities listed in the Chosen Application section.

Laravel is described as a “Web Ecosystem” for the PHP language, containing package managers, products, kits and services that aid in development, such as easy database integration with a range of databases, monitoring tools for software performance and testing frameworks (Laravel, 2025).

## Directory Structure

The starting point for the application has a strange and unintuitive design structure, loosely separated out by functionality, with each major sub-class having a folder associated with it. Most modern applications have a source folder, where all of the data is then separated into each category.

The approach taken by the application is to divide out the code into sub-categories, such as a folder called “updateCustomer”, when a more sane approach would be to divide the categories into major categories first, then the sub-categories, representing a more conventional, tree-like structure, with a place for everything, and everything in its place.

Running the Windows ‘Tree’ command from the source folder will produce the projects structure for viewing. The following page holds the result of that command:

Source Directory/

|

| .gitignore

| header.html

| LICENSE

| README.md

| shippingfile.sql

| styles.css

| T00158237\_Daniel\_Jameson\_Cloud\_Project.docx

| TreeStructure.txt (This file)

|

+------addCustomer

| addCustomer.html

| addCustomer.php

|

+------addGame

| addGame.html

| addGame.php

|

+------images

| favicon.ico

| img1.jpg

| img2.jpg

|

+------login

| login.html

| login.php

|

+------orders

| manageOrders.php

| placeOrder.php

|

+------updateCustomer

| delete.php

| deletecustomer.php

| header.html

| selectupdate.php

| update.txt

| updated.php

| updatedetails.html

| updateform.php

| view all update delete.php

| viewUpdateDelete.php

| whotoupdate.html

|

+------updateGame

delete.php

deleteGame.php

updated.php

updateDetails.html

updateForm.php

viewUpdateDelete.php

## Clean Code

The application code is very messy, having no obvious indentation or order of operations as a result. In the old version, it is difficult to gather where one code block begins and the other ends, as seen in the following blocks of code: A screen shot of a computer program

AI-generated content may be incorrect.

In the page above, the while loop starting inside the if condition has no obvious starting and finishing point, as it does not end where the indentation for the angled brackets begin. Add to this that the code within the try statement is negatively indented, making the code harder than necessary to read, leading to problems with debugging and merging code.

# Glossary

AWS(Amazon Web Services) – A series of services available for cloud hosting on Amazon.

PHP (Hypertext Pre-Processor) – Programming language designed to run on Server-Side applications.

Laravel – Framework designed around the PHP language.

MySQL – A Structured Query Language relational database technology that is open source. It is commonly associated with the languages used in this project.

Xampp – A basic Server that is used to deploy web applications, owned and maintained by Apache.

# References