February 11, 2021

Winter Semester 2020/21

Advanced Management of Data

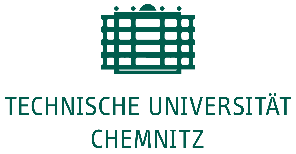
Project Term Paper

**SUBMITTED BY**

|  |  |
| --- | --- |
| Name : Mir Sohrab Ali | Name : Md Abdul Awal |
| Matriculation # 621531 | Matriculation # 614638 |
| Study Program: Automotive Software Engineering | |

PROJECT NAME

**PizzaBin**

PizzaBin is a project which is a task for the final exam of Advanced Management & Data. This project is similar to a pizza management system where users can choose a pizza with varieties of ingredients and give orders to a pizza baker. Pizza baker manages ingredients, suppliers & pizzas.

Contents

[1 Introduction 1](#_Toc63957272)

[2 Project: PizzaBin 1](#_Toc63957273)

[2.1 Requirements 1](#_Toc63957274)

[2.1.1 Main Requirements Breakdown & Task Distribution 2](#_Toc63957275)

[2.1.2 Additional Requirements Breakdown & Task Distribution 3](#_Toc63957276)

[2.2 Implementation 4](#_Toc63957277)

[2.2.1 Database 4](#_Toc63957278)

[2.2.2 Application Architecture 6](#_Toc63957279)

[2.2.3 Functionality Explanation 6](#_Toc63957280)

[2.2.4 DB Functions invoking list 10](#_Toc63957281)

[2.2.5 Sample Screenshots 11](#_Toc63957282)

[2.3 Future Development Scope 12](#_Toc63957283)

[2.3.1 Distributed Database 12](#_Toc63957284)

[2.4 Deployment 12](#_Toc63957285)

[2.4.1 Package Overview 12](#_Toc63957286)

[2.4.2 Installation and Prerequisites 12](#_Toc63957287)

[2.4.3 Configure 13](#_Toc63957288)

[2.4.4 How to Run 13](#_Toc63957289)

[3 Technology Overview 13](#_Toc63957290)

[3.1 PHP 13](#_Toc63957291)

[3.2 jQuery 14](#_Toc63957292)

[4 Conclusion 15](#_Toc63957293)

**References ……………………………………………………………………………………………………… 16**

Figures

[Figure-1: Project Structure [1] 1](#_Toc63957258)

[Figure-2: Relational Schema 4](#_Toc63957259)

[Figure-3: UML Diagram 5](#_Toc63957260)

[Figure-4: Application Architecture 6](#_Toc63957261)

[Figure-5: UI - Ingredient Settings Screenshot 7](#_Toc63957262)

[Figure-6: Script of displaying ingredient settings UI 7](#_Toc63957263)

[Figure-7: Script of handling post data from ingredient settings 8](#_Toc63957264)

[Figure-8: Script of calling db function 9](#_Toc63957265)

[Figure-9: Pizza Selection (Home page) 11](#_Toc63957266)

[Figure-10: Ingredient Selection 11](#_Toc63957267)

Tables

[Table-1: Main requirements breakdown 3](#_Toc63957268)

[Table-2: Additional requirements breakdown 4](#_Toc63957269)

[Table-3: DB Function invoking list 10](#_Toc63957270)

[Table-4: Required Package for Installation. 13](#_Toc63957271)

# Introduction

The aim of the project is to develop a pizza management system for achieving the final examination requirement of “Advanced management of Data”. This will help make order entry easier, make customers happier and help build a good business and through this project, we will demonstrate our learned skill of “Advanced management of Data”.

# Project: PizzaBin

## Requirements

The main requirement of this project is to build the project using PostgreSQL database where program logic implemented in PL/pgSQL language. Also, A frontend for user interaction with the data from the database. The diagram below shows how the frontend will connect and communicate with the database. Later we will see in detail how this architecture works.

Frontend

Database

Figure-1: Project Structure [1]

### Main Requirements Breakdown & Task Distribution

The table below illustrates the main requirements in more detail.

|  |  |  |  |
| --- | --- | --- | --- |
| Application STACK | Task | Remarks | Task Distribution |
| Frontend | UI for customer view (default view) |  | Awal |
| UI for base pizza selection | Customer order will be initiated by this selection. | Awal |
| UI for ingredient selection for composing with the base pizza | Show name, regional province & price. Only active and available stock’s ingredients will display. | Ali |
| UI for baker view | Switch on user selection | Awal |
| Show list of recent orders in baker view as default |  | Ali |
| UI for supplier management | Add, change, hide/show, remove | Awal |
| UI for ingredients management | Add, change, hide/show, remove | Awal |
| UI for setting ingredient price and assigning regional province. |  | Ali |
| UI for ingredients restock | Only available supplier will be displayed with immediate restocking | Awal |
| Database | Create relational schema |  | Awal, Ali |
| Create UML diagram for tables |  | Ali |
| Create tables |  | Awal, Ali |
| Backend | Functions for fetching base pizza list |  | Ali |
| Function for storing user selected pizza | Basic order information also | Ali |
| Function for fetching ingredient list |  | Awal |
| Function for storing selected ingredient pizza composition |  | Ali |
| Function for confirming order |  | Awal |
| Function for fetching recent order list |  | Awal |
| Functions for storing , editing and removing supplier |  | Ali |
| Functions for storing , editing and removing ingredients |  | Ali |
| Functions for fetching ingredients & price for each regional province. |  | Awal |
| Functions for fetching ingredients, province, supplier for restocking |  | Ali |
| Function for storing restock request |  | Awal |
| Function for showing current stock. |  | Awal |

Table-1: Main requirements breakdown

### Additional Requirements Breakdown & Task Distribution

The table below illustrates additional requirements that was found necessary to achieve the main requirement.

|  |  |  |  |
| --- | --- | --- | --- |
| Application STACK | Task | Remarks | Task Distribution |
| Frontend | UI for pizza settings | Add, edit, remove | Awal |
| UI for regional province settings | Add, edit, remove | Ali |
| UI for Base Pizza Price Settings | Set price for all sizes pizzas | Awal |
| UI for selecting ingredients for each individual supplier. | Specifies which supplier can supply which ingredient. | Ali |
| Backend | Functions for storing, removing & fetching pizza |  | Awal |
| Functions for storing, removing & fetching regional province |  | Awal |
| Functions for fetching base pizza & sizes |  | Ali |
| Functions for storing prices of all sized pizzas |  | Awal |
| Functions for fetching active suppliers |  | Awal |
|  | Functions for fetching ingredients with province | Only active | Ali |

Table-2: Additional requirements breakdown

## Implementation

### Database

#### Relational Schema

Relational schema of a database defines the design and structure of the relation like it consists of the relation name, set of attributes/field names/column names. Every attribute would have an associated domain [4]. Here we defined the primary key by indicating underline and **foreign key** by making the font bold.

Pizza (pizza\_id, name, image\_path, isActive)

Pizza\_size (size\_id, size\_name, dia\_inch, isActive)

Pizza\_price (**size\_id**, **pizza\_id**, price)

Ingrediant\_backer\_stock (**ing\_prov\_id**, qty)

Ingrediants (ingrediant\_id, Name, isActive)

Ingrediant\_province (ing\_prov\_id, **ingrediant\_id**, **province\_id**, price, isActive)

Regional\_Province (province\_id, Name, isActive)

Suppliers (supplier\_id, Name, isActive, Address, Contact\_no, Email)

supplier\_ingrediants (**supplier\_id**, **ing\_prov\_id**, sup\_ing\_ref\_id)

Order\_status (status\_id, status)

Order (order\_id, order\_date\_time, total\_bill, isPaid, **status\_id**)

order\_pizza (**order\_id**,**pizza\_id**, **size\_id**, order\_composition\_id)

order\_pizza\_ingrediants (id, **order\_composition\_id**, **ing\_prov\_id**, qty)

Figure-2: Relational Schema

Relational schema has a very important role for developing the project. First we did this relational schema even before starting to write program. Because relational schema helps us to understand how the tables of the db are connected to each other using foreign key and primary key.

#### UML Diagram

The UML (Unified Modeling Language) is general-purpose modeling language which provides a visual representation of a database [5]. For this project, we UML diagram was necessary to specify the structure or behavior of system. Moreover, it allows us to visualize this system therefore, it helps us to understand the complexity of this project part by part.

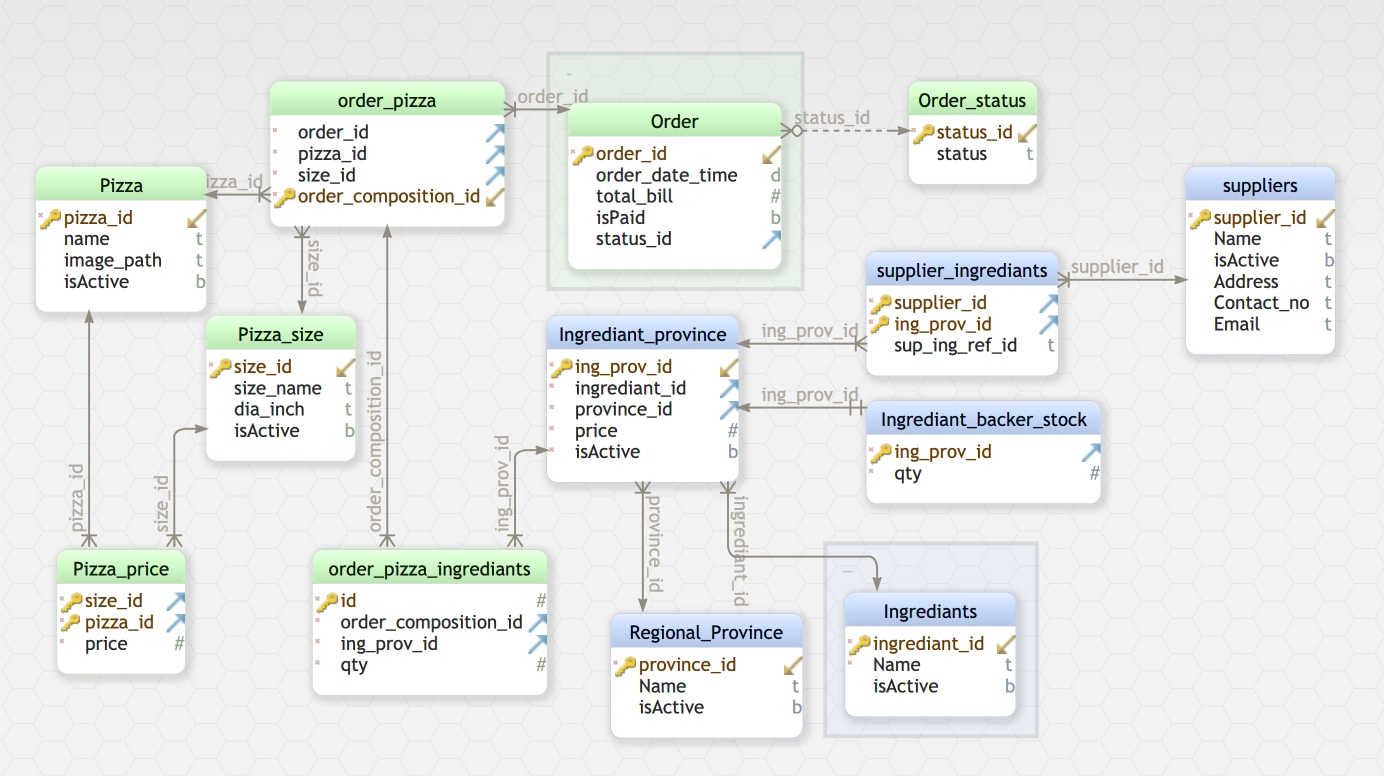


Figure-3: UML Diagram

The Figure-3 represents the UML diagram of our database and illustrates how the tables are connected as which relations like one to many or many to one relationship. Not only our goal was to fulfill the requirements of the tasks but also we gave the priority to normalize the database as much as possible.

### Application Architecture

Post Data Handler

DB Controller

PHP

**PostgreSQL Database**

Tables

PL/pgSQL

User Interface

**Frontend**

Figure-4: Application Architecture

The overall project has been broken down into two separate structure whose are frontend and database respectively.

**Frontend**

The frontend consists of 2 layers:

* **User Interface:** This layer handles all user interactions as well as displays data from the database. We have used HTML, CSS, Bootstrap, Javascript, and Ajax.
* **Server side scripting (PHP):** This layer communicates with the user interface and database. We have used PHP as the scripting language. PHP handles two kinds of operations. One is it can receive posted data given by a form and it can send that posted data to a server.

**Database**

All of our data will be stored in a database. As per requirement, we used PostgreSQL DB. Our logic of handling the database is implementing via PL/pgSQL which is inside the database.

### Functionality Explanation

This section will show how a complete functionality is executed. As an example here we will show how the UI of ingredient settings is showing and how the user data will store in the database.

**Users Input:** In the following figure illustrates, user can add an ingredient by clicking the plus (+) icon, change the ingredients name, set the status as active or inactive (hide/show) and delete any ingredient.

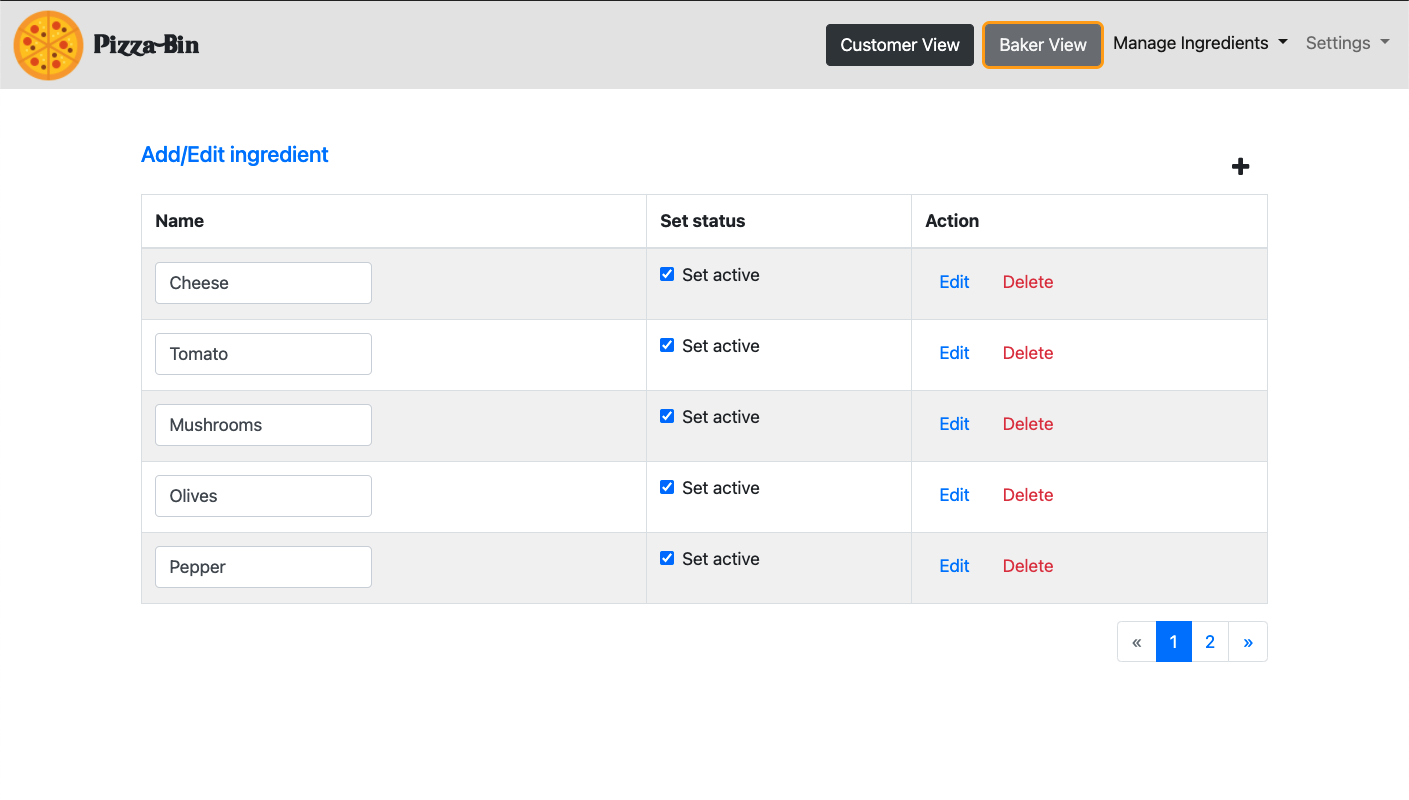
****

Figure-5: UI - Ingredient Settings Screenshot

**Displaying UI**

By using html, css embedded with php, we displayed the above UI. In this example, we sent the given user data as POST data by submitting a form.

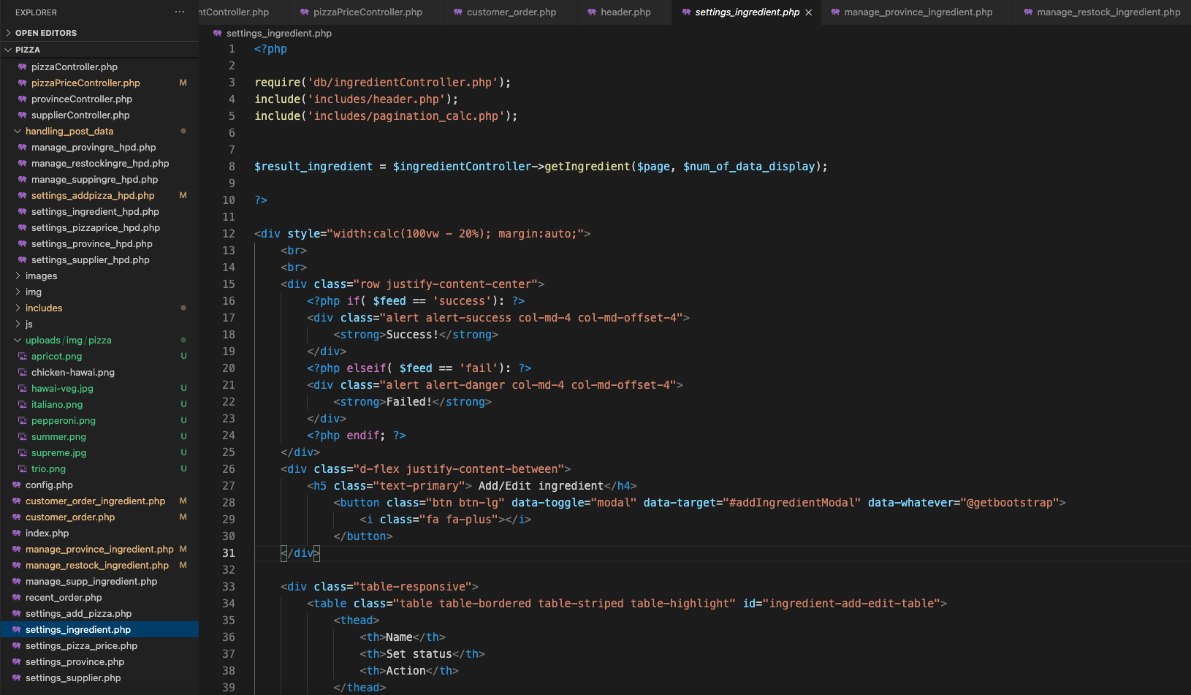
****

Figure-6: Script of displaying ingredient settings UI

**Handling POST data**

This script demonstrates how the post data is received and sent to the DB by calling our custom method (saveIngredient()).

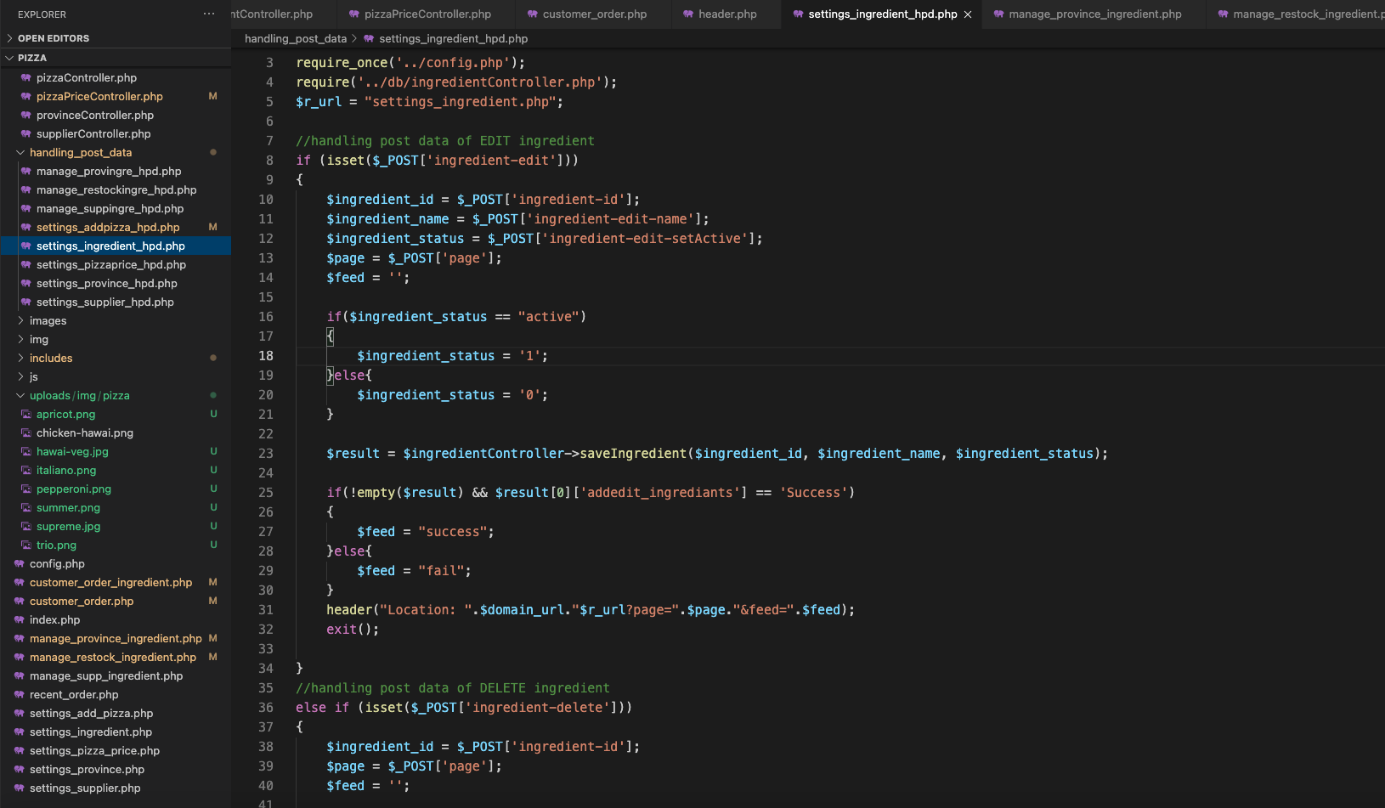
****

Figure-7: Script of handling post data from ingredient settings

**Calling DB function**

Our custom method “saveIngredient()” passes the POST data to the db controller and db controller sends the POST data as arguments by calling DB function.

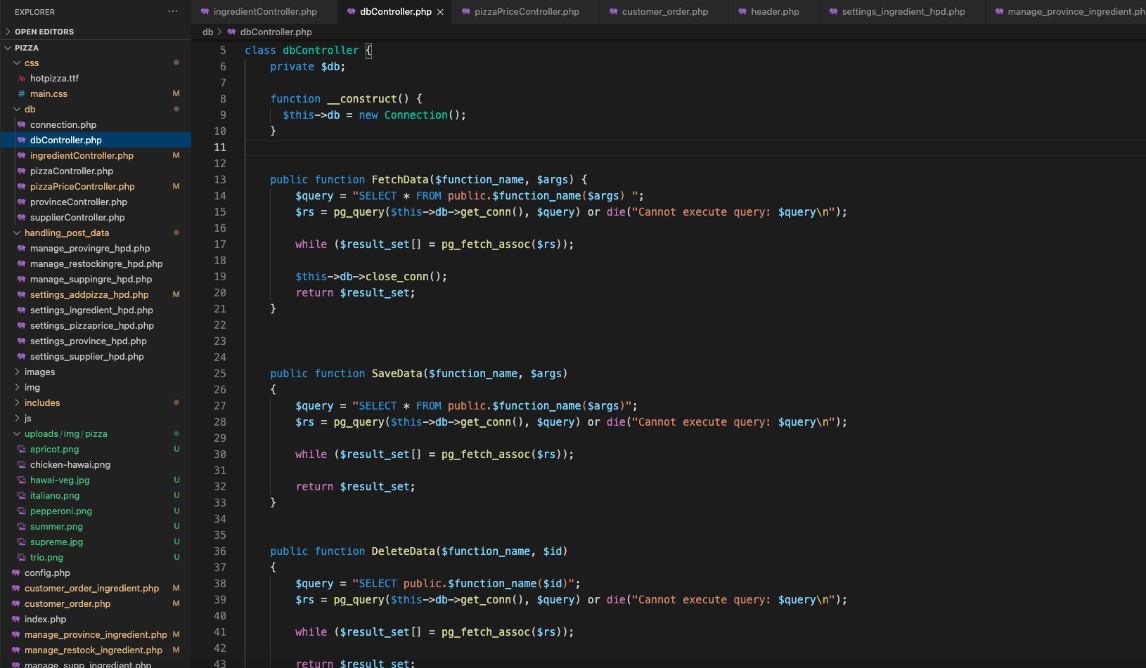
****

Figure-8: Script of calling db function

**SQL of DB Function**

Our predefined db function stores the given data to the database.

|  |
| --- |
| *-- FUNCTION: public.addedit\_ingrediants(integer, text, bit)*    *-- DROP FUNCTION public.addedit\_ingrediants(integer, text, bit);*    **CREATE** **OR** **REPLACE** **FUNCTION** public.addedit\_ingrediants(  v\_id **INTEGER**,  v\_name **TEXT**,  v\_active **BIT**)  **RETURNS** **TEXT**  **LANGUAGE** 'plpgsql'  **COST** 100  **VOLATILE** PARALLEL UNSAFE  **AS** $BODY$  **BEGIN**    **IF** (v\_id<0) **THEN**  **INSERT** **INTO** public."Ingrediants"(  "Name", "isActive")  **VALUES** ("v\_name", "v\_active");  **ELSE**    **IF** **NOT** EXISTS(**SELECT** 1 **FROM** public."Ingrediants"  **WHERE** "ingrediant\_id"="v\_id") **THEN**  **RETURN** 'No record found';  **ELSE**  **UPDATE** public."Ingrediants"  **SET** "Name"="v\_name", "isActive"="v\_active"  **WHERE** "ingrediant\_id"="v\_id";  **END** **IF**;  **END** **IF**;    **RETURN** 'Success';    exception  **WHEN** OTHERS **THEN**  raise exception 'Failed';  **END**;  $BODY$;    **ALTER** **FUNCTION** public.addedit\_ingrediants(**INTEGER**, **TEXT**, **BIT**)  **OWNER** **TO** pizzaamd\_rw; |

Source Code: Functions of Ingredient Settings

### DB Functions invoking list

|  |  |  |
| --- | --- | --- |
| **Frontend** | **DB- Functions-Invoked** | **Tables** |
| Pizza Settings | get\_all\_pizzas() | Pizza |
| addedit\_pizza() |
| remove\_pizza() |
| Ingredient Settings | get\_all\_ingrediants() | Ingrediants |
| addedit\_ingrediants() |
| remove\_ingredient() |
| Regional Province Settings | get\_all\_provinces() | Regional\_Province |
| addedit\_province() |
| remove\_province() |
| Supplier settings | get\_all\_suppliers() | suppliers |
| addedit\_supplier() |
| remove\_supplier() |
| Pizza price settings | get\_pizza\_price\_list() | Pizza\_price, Pizza, Pizza\_size |
| addedit\_pizza\_price() | Pizza\_price |
| Manage Ingredient by supplier | get\_active\_suppliers() | suppliers |
| get\_ingredients\_by\_suppliers() | supplier\_ingrediants,  suppliers, Ingrediants |
| addedit\_ingredient\_of\_supplier() | supplier\_ingrediants |
| Ingredient price by province | get\_active\_ingrediants() | Ingrediants |
| get\_provinces\_n\_price\_by\_ingredient() | Ingrediants, Regional\_Province, Ingrediant\_province |
| addedit\_provinces\_of\_ingredient() | Ingrediant\_province |
| Restock | get\_provinces\_by\_ingredient() | Ingrediant\_province, Regional\_Province |
| get\_suppliers\_by\_ingredient\_province() | supplier\_ingrediants, suppliers |
| restock\_ingrediant() | Ingrediant\_backer\_stock |
| get\_stocks() | Ingrediant\_backer\_stock, Ingrediant\_province, Ingrediants, Regional\_Province |
| Recent Orders | get\_recent\_orders() | order\_pizza, Pizza, Pizza\_size, Order, Order\_status |
| Order Pizza -> Base pizza selection | add\_order\_pizza() | order\_pizza |
| add\_order\_pizza\_ingredients() | order\_pizza\_ingrediants |
| Order Pizza -> Ingredient selection | get\_ingredient\_price\_list() | Ingrediant\_province, Ingrediants, Regional\_Province |
| addedit\_order() | Order |

Table-3: DB Function invoking list

### Sample Screenshots

|  |
| --- |
| Figure-9: Pizza Selection (Home page)    Figure-10: Ingredient Selection Future Development ScopeDistributed Database We can use distributed database on following scenario. Multiple backer shops locating at different geographical locations. When the backer shop will be subdivided into multiple branches that will be physically distributed over the globe. Each unit would requires its own set of local data. In overall database of the organization may require distributed. |

## Deployment

### Package Overview

The application is distributed as a zip archive (pizzabin.zip).

pizzabin

|\_\_\_ **app** [ frontend ]

| |\_\_\_ README.txt

| |\_\_\_ index.php

| |\_\_\_ includes

| |\_\_\_ other artifacts …

|

|\_\_\_ **dbfiles**  [ database ]

| |\_\_\_ README.txt

| |\_\_\_ pizzabin.tar

| |\_\_\_ pizzabin.sql

| |\_\_\_ other artifacts …

### Installation and Prerequisites

The zip file needs to extract at the server’s htdocs directory (for local server) or public\_html directory (for remote server).

|  |  |
| --- | --- |
| APPS | REQUIRED PACKAGES |
| App | PHP==7 or above  Bootstrap==4  Local Server==XAMPP (Windows) or MAMP (Mac) |
| Database | PostgreSQL = 11  Local postgreSQL client: pgAdmin4 |

Table-4: Required Package for Installation.

### Configure

We have created our database locally and we tested our script on TUC server. Here is an example configuration for that:

1. Move the app folder to a local server's directory (htdocs).

2. Edit the \_\_construct function of the app/db/connection.php file as bellow example

**Configuration Parameters:**

host = pgsql.hrz.tu-chemnitz.de

port =5432

db\_name = pizzaamd

user = pizzaamd

pass = iujae4Daegh9

### How to Run

After starting the server (XAMPP/MAMP), we just need to hit the URL (directory of our app at the server) at the browser. In our environment the URL was like:

|  |
| --- |
| http://localhost:8888/pizzabin |

That’s it. Now the application will run and ready for showing the functionalities of a pizza management system.

# Technology Overview

## PHP

PHP-logo.svgPHP is a general-purpose scripting language especially suited to web development. It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994. The PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control Arbitrary PHP code can also be interpreted and executed via command-line interface (CLI).

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge

The PHP language evolved without a written formal specification or standard until 2014, with the original implementation acting as the de facto standard which other implementations aimed to follow. Since 2014, work has gone on to create a formal PHP specification.

As of January 2021, 72% of PHP websites use discontinued versions of PHP, i.e. PHP 7.2 or lower, which are no longer supported by The PHP Development Team. A large additional fraction uses PHP 7.3, which is only (up to December 6, 2021) "supported for critical security issues only. Over 40% of all PHP websites use version 5.6 or older that not even Debian supports (Debian 9 supported version 7.0 and 7.1) [3].

## jQuery · GitHubjQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript [6].

In this project we used jquery along with ajax for generating a pop up window for executing some baker’s functionality.

# Conclusion

The overall development experience of PizzaBin (a pizza management system) was absolutely satisfying. Especially the opportunity to explore the PostgreSQL database. We tried to build the database in a way that delivers most efficient results. In addition, we looked after the maximum normalization possible to achieve along with full filling all the required tasks. We designed the database table in a way that the application can be scalable and some real world features can also be possible to implement. We tried to provide a user-friendly UI and better experience and last but not the least this pizza management system satisfies all the given tasks and ready for future implementation.

**References**

1. Daniel Richter (2021. January 7) “AMD Project Task” [PDF File]. Retrieved from <https://www.tu-chemnitz.de/informatik/DVS/lehre/AMD/Project/AMD%20-%20Project.pdf>
2. Daniel Richter (2021. January 7) “Questions and Answers concerning project task in AMD Project Task” [Text File]. Retrieved from <https://www.tu-chemnitz.de/informatik/DVS/lehre/AMD/Project/AMD%20-%20Project%20QA.txt>
3. Wikipedia(2021. January 25). Retrieved from [https://en.wikipedia.org/wiki/PHP](https://en.wikipedia.org/wiki/PHP%20)
4. Geeksforgeeks(2021. February 08). Retrieved from <https://www.geeksforgeeks.org/relation-schema-in-dbms/>
5. Wikipedia(2021. February 08). Retrieved from <https://en.wikipedia.org/wiki/Unified_Modeling_Language>
6. jQuery. (2021, Febri). Retrieved from <https://jquery.com/>