Andrew Nguyen - 1478375
Shahenshah Meghani - 1625080
Syed Aziz - 1623052

Introduction

What did we use to make our website?

For the frontend we used HTML and CSS to create the interface that users will interact with. We used it for our project since it was easy to learn and many of us already had previous experience in coding it.

For the backend we used PHP as our programming language to give functionality to our website. Originally we were trying to use python, but during development we switched over to PHP because we found it to be much easier to use. It is also very compatible with WAMP Server.

For our database we used MySQL to store user information on our website. Many of us have experience with PostgreSQL and MySQL was very similar to that. However, WAMP Server had built in support for MySQL so we decided to use that instead of PostgreSQL.

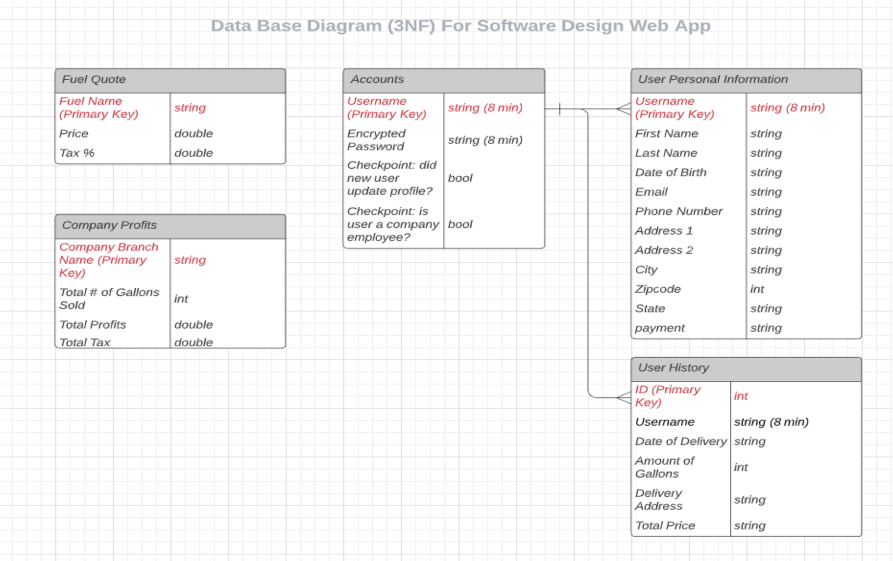
How did we build our website?

We used WAMP Server to help construct and put our website together. We used WAMP Server because it was very easy to use and it had built in support for both MySQL and PHP.

To use WAMP Server we put our HTML/CSS and PHP code (all in a folder called A3) into the www file directory. After launching WAMP Server on the desktop, you can access login page by going to URL: http://localhost/A3/

Talk more about your Database?

Our Database has a total of five tables, and all code used to make it is located within the file MyDataBase.sql on github. The goal of designing our database was to keep it as simple as possible and create all our tables in a way to reach 3rd normal form. This means that our Database does not contain any transitive partial dependency and ensures referential integrity. Below you can see a diagram of all the tables, note that each table has a primary key highlighted in red.



Data Base Diagram (3NF) For Software Design Web App

Fuel Quote	
Fuel Name (Primary Key)	string
Price	double
Tax %	double

Company Profits	
Company Branch Name (Primary Key)	string
Total # of Gallons Sold	int
Total Profits	double
Total Tax	double

Accounts	
Username (Primary Key)	string (8 min)
Encrypted Password	string (8 min)
Checkpoint: did new user update profile?	bool
Checkpoint: is user a company employee?	bool

User Personal Information	
Username (Primary Key)	string (8 min)
First Name	string
Last Name	string
Date of Birth	string
Email	string
Phone Number	string
Address 1	string
Address 2	string
City	string
Zipcode	int
State	string
payment	string

User History	
ID (Primary Key)	int
Username	string (8 min)
Date of Delivery	string
Amount of Gallons	int
Delivery Address	string
Total Price	string

Actual Demo

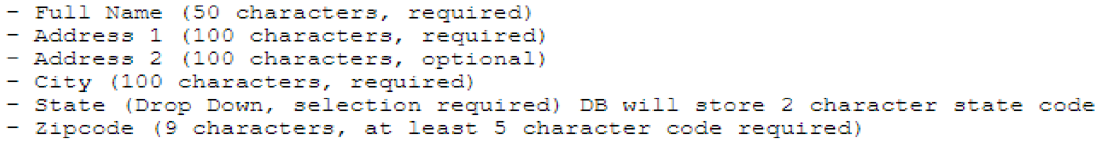
Login Page:

- Show already in db username and password works, but fake does not.
- Min of 8 char needed for both username and pass.
- We use openssl_decrypt() Function to get real password from DB, show sql.
- Show log off.

Show Create Account:

- Show using prev username gives error.
- Show Cancel
- Make real user with password 87654321
- We use openssl_encrypt() Function to store password to DB, show sql.

Update Profile:



```
- Full Name (50 characters, required)
- Address 1 (100 characters, required)
- Address 2 (100 characters, optional)
- City (100 characters, required)
- State (Drop Down, selection required) DB will store 2 character state code
- Zipcode (9 characters, at least 5 character code required)
```

- Show all text boxes.
- Explain that cancellation is dynamic, show 1st cancel.
- Fill info, set to texas and fill all addresses, show sql table personal.
- show 2nd cancel.
- Fill again, set to texas and only one address, show sql table personal.

Buy Fuel:

- Create additional accounts with user not in texas.
- Show no history.
- Gallons can’t be negative, must choose a valid address.
- Margin = Current Price * (Location Factor - Rate History Factor + Gallons Requested Factor + Company Profit Factor)
- Texas, No His, low fuel (100) → suggested = 1.725
- Texas, No His, high fuel (1500) → suggested = 1.71
- Texas, His, low fuel (100) → suggested = 1.71
- Texas, His, high fuel (1500) → suggested = 1.695
- Not Texas, No His, low fuel (100) → suggested = 1.755
- Not Texas, No His, high fuel (1500) → suggested = 1.74
- Not Texas, His, low fuel (100) → suggested = 1.74
- Not Texas, His, high fuel (1500) → suggested = 1.725
- Text boxes can't be manipulated.
- cancel work.

History:

- History is dynamic
- Show sql table history
- Go back works

Company Info:

- can only be used by employee user “username”, otherwise display error and go to login
- Show sql table

Unit Testing

What did we use to make unit tests for our program?

We used PHPUnit with Xdebug (the one that came installed with WAMP Server) to make and run tests on our classes.

We followed the video below to help us get our testing environment running: https://www.youtube.com/watch?v=cQucsmbOkpE&list=PLf7SQrI5f5j7fy3c6ekaKCIPUd51u-CcA&index=4&t=11s

How do you run your tests?

1) PHP with Xdebug and PHPUnit must be installed on the computer.
2) We stored our tests and classes into the file directory called tests.
3) Go to PHPUnit directory in Command Prompt → cd C:\Users\Shahenshah Meghani\phpunit
4) Run Command → .\vendor\bin\phpunit → to run unit tests on classes.

About our unit tests?

We have a total of 30 tests with 30 assertions, and we achieve 100% code coverage.

We tested a slightly modified version of the classes that is currently used within our website. The reason for these changes is because we needed hard coded values that represent database content. But despite these slight differences the classes are basically the same, and these tests helped us a lot when building our website.

These tests were designed by testing class functionality such as: calculating correct fuel quote based on specific user information, redirecting the user to the correct html pages, being able to reach the end of every function within all classes, and etc.