# INTRODUCTION

Energy is a vital issue in the modern world. At present, the machine is part of parcel of the daily life. Moreover, most of the machines are depending on energy or fuel in order to operate. The modern life depends on energy and most of the energy is coming from Petroleum or natural resources. Most of the countries in the world have not available petroleum or other natural resources. They have some small amount of natural Gas mine which is not good enough for a long time. Maximum developing country's government imports a large amount of petroleum in every year and sells them at fewer prices than the international market. The government of Bangladesh gives a large amount of subsidy in petroleum sector every year. The government of Bangladesh spends a major share of its budget providing direct subsidies for fossil fuels and electricity, the costs of which have been escalating rapidly in recent years. In 2012, the government reports it spent BDT 81.4 billion (US$ 944 million) in direct expenditure on energy subsidies [1]. In 2014 the finance ministry has sanctioned BDT 1,253 core in favor of the Bangladesh Petroleum Corporation as subsidy payments for selling petroleum products in the domestic market at prices lower than its procurement costs [2]. A well monitoring is necessary in this sector. The idea is to develop such a system that can monitor every transaction of every fuel station (e.g. Petrol or CNG pump). The central goal of this research is to automate the fuel station so that the transactions can be maintained without difficulty.

## 1.1. Organization Profile

Currently, we are dealing with three major start-ups built by us. All started with our first start up Retech Projects which provides service and supports for engineering students in their engineering projects and course internships. It is responsible for all the research activities carried by our all other companies. Retech Lasers is created to manufacturing machinery which uses laser technology in their core. We have just started working simultaneously on BLDC motor design and development which is delta by one of our start up called Retech Motors. We are interested to create opportunities for young graduates. Feel free to reach to take part in our mission.

**Website**

[**http://www.retechsolutions.in**](https://www.linkedin.com/redir/redirect?url=http%3A%2F%2Fwww%2Eretechsolutions%2Ein&urlhash=Yx-H&trk=about_website)

**Industries**

Design Services

**Company size**

11-50 employees

**Headquarters**

Chennai, Tamil Nadu

**Type**

Privately Held

## System Specifications

### HARDWARE CONFIGURATION

**Processor** : Pentium -IV

**Speed** : 1 GHz

**Hard Disk Capacity** : 40GB

**RAM Capacity** : 1GB RAM

**CD-ROM Drive** : 52x speed

**Keyboard** : 104 keys

**Mouse** : Logitech

**Printer** : HP3745 series DeskJet printer

### SOFTWARE SPECIFICATION

**Operating System** : Windows 7/8/10

**Front End** : JAVA

**Back End** : SQL

**Feasibility Study**

# SYSTEM STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility of a proposed solution is evaluated in teams of its components. These components are:

* + - * Economic feasibility
      * Technical feasibility

## Economic Feasibility

The economic feasibility study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development or the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## Technical Feasibility

The technical feasibility study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The will lead to high demands on the available technical resources. This will lead to high demands being places on the client. The developed system must have modest requirements, as only minimal or null changes are required for implementing this system.

## EXISTING SYSTEM

In present system all report work is done manually at the petrol pump. Admin user have to check each petrol pump machine the calculate the sold petrol, but in our proposed system user can check sold petrol from his monitor by clicking on some report button. These systems also keep records of each employee. Admin can also check which employee sold how much petrol and what amount he submitted to accountant at the end of the day. Admin can also check how much oil they sold in one day or one month. Admin can generate monthly sale report or annually sale report when they want. These systems also help admin to generate the salary of each employee on the basis of his position and working hour.

### DRAWBACKS

* Maintenance costs: Like any technological system, fuel automation systems require regular maintenance and updates to ensure they are functioning correctly. These costs can add up over time.
* Limited compatibility: Some fuel automation systems may not be compatible with all types of fuel tanks, which can limit the options available for fuel retailers.

## PROPOSED SYSTEM

As the work at petrol pump is done manually so it becomes hard for the supervisor to maintain a daily record. So we have developed this system to computerize all the reporting work of petrol pump. Our project gives the supervisor to maintain a daily record in an easiest way. Admin can add new employee and can also delete employee from the system. this system also keep track of the employee attendance and generate the salary on the basis of his working hours. Employee can also check the petrol diesel price online. Admin can also find contact detail of every employee from the system. he can find detail like his address , phone number , joining date , native address.

From report module admin can also generate several reports like pumps daily earning report , employee shift delivery report , pumps report , cumulative daily report , salary report. Admin can update many information in the system that’s why his account is protected with user name and password. Only authentic user can login to admin account

### FEATURES

* Time and cost savings: Fuel automation systems reduce the need for manual tank level readings, which can save time and labor costs for fuel retailers.

# SYSTEM DESIGN AND DEVELOPMENT

Design is concerned with identifying software components specifying relationship Among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is Minimal clearly specified. Design will explain software components in details. This will help the implementation of the system. Moreover, this will guide the further changes in the system to satisfy the further requirements.

The design document describes how to transform, the requirement and the functional design into more technical system design specification. This design involves conceiving and planning out in the mind and making a drawing pattern of sketch of. It includes type of activities, External Design, Architectural Design and Detailed Design. The architectural design and detailed design collectively referred to as internal design.

The external design involves specifying the externally observable characteristics of a software product and the internal design involves specifying the internal structure and processing details of the system. The fundamental concept of the design includes abstraction structure, information hiding Modularity, concurrency, verification and design aesthetics.

## FILE DESIGN

In computing, a file design (or file system) is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names are called a "file system".

Some file systems are used on local data storage devices; others provide file access via a network protocol. Some file systems are "virtual", in that the "files" supplied are computed on request or are merely a mapping into a different file system used as a backing store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

## INPUT DESIGN

The input design is the process of entering data to the system. The input design goal is to enter to the computer as accurate as possible. Here inputs are designed effectively so that errors made by the operations are minimized.

The inputs to the system have been designed in such a way that manual forms and the inputs are coordinated where the data elements are common to the source document and to the input. The input is acceptable and understandable by the users who are using it.

Input design is the process of converting user-originated inputs to a computer-based format input data are collected and organized into group of similar data. Once identified, appropriate input media are selected for processing.

The input design also determines the user to interact efficiently with the system. Input design is a part of overall system design that requires special attention because it is the common source for data processing error. The goal of designing input data is to make entry easy and free from errors.

Input design is the process of connecting the user-originated inputs into a computer to used format.

The goal of the input design is to make the data entry logical & free from errors.

## OUTPUT DESIGN

Output design is the process of converting computer data into hard copy that is understood by all. The various outputs have been designed in such a way that they represent the same format that the office and management used to.

Computer output is the most important and direct source of information to the user. Efficient, intelligible output design should improve the systems relationships with the user and help in decision making. A major form of output is the hardcopy from the printer.

Output requirements are designed during system analysis. A good starting point for the output design is the Data Flow Diagram (DFD). Human factors educe issues for design involves addressing internal controls to ensure readability.

The output form in the system is either by screen or by hard copies. Output design aims at communicating the results of the processing of the users. The reports are generated to suit the needs of the users. The reports have to be generated with appropriate levels.

All reports are output formats, maintained details can be reported over crystal reports, this project sustain following reports

## DATABASE DESIGN

The most important consideration in designing the database is how information will be used.

The main objectives of designing a database are:

### Data Integration

In a database, information from several files are coordinated, accessed and operated upon as through it is in a single file. Logically, the information are centralized, physically, the data may be located on different devices, connected through data communication facilities.

### Data Integrity

Data integrity means storing all data in one place only and how each application to access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications, which use it. This leads to less data redundancy; data items need not be duplicated; a reduction in the direct access storage requirement.

### Data Independence

Data independence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming of applications and to allow modifications to application programs without reorganizing the physical data.

The tables needed for each module were designed and the specification of each and every column was given based on the records and details collected during record specification of the system study.

## SYSTEM DEVELOPMENT

The key to control maintenance costs is to design systems that are easy to change, so the link between development and maintenance is very strong. Many of the analysis and design methodologies, tools, and techniques employed during system development can be applied to system maintenance, but there are significant differences between development and maintenance. Maintainability is the ease with which software can be understood, corrected, adopted and enhanced.

### DESCRIPTION OF MODULES

To develop this project several step should be followed. There are various modules in this proposed system they are listed below.

* + - * Admin
      * Customer
      * Maintenance
      * Sales
      * Balances

### Admin:

Admin has all the access to manage this application, but admin should enter correct username and password to access this application. If the admin given invalid username and password they won’t allow to access this application.

### Customer:

Admin click the customer module and managing the customer details. Admin have rights to add and manage the employees details. Admin should enter customer name, email, contact and address details. Also admin can have rights to edit and delete the customer details.

### Maintenance:

This module has all the details about the fuel and diesel brand and price details. can modify and delete the records. Once it is created it will be stored in the maintenance table whenever data is needed can get it from the table.

### Sales:

This module could be select the fuel type and price to calculate the price details. This module has to manage all the performance activity. Also we can select the customer and reduce the balance amount.

### Balance:

This module can be perform to calculate the amount on customer level, if the customer has any pending payment amount they can manually choose and pay the amount.

# TESTING AND IMPLEMENTATION

System testing is the process of exercising software with the intent of finding and ultimately correcting errors. This fundamental philosophy does not change for web applications, because Web-based systems and application reside on a network and interoperate with many different operating system, browsers, hardware platforms, and communication protocols; the search for errors represents a significant challenge for web application.

The distributed nature of client\server environments, the performance issues associated with transaction processing, the potential presence of a number of different hardware platforms, the complexities of network communication, the need to serve multiple clients from a centralized database and the requirements imposed on the server all combine to make testing of client\server architectures.

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. System testing is the state of implementation that is aimed at assuring that the system works accurately and efficiently. Testing is the vital to the success of the system. System testing makes the logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

### Unit Testing

Unit testing focuses verification efforts on the smallest unit of software design of the module. This is also known as “module testing”. This testing is carried out during programming stage itself. In this testing step, each module is found to be working satisfactorily as regards to the expected output of the modules.

### Integration Testing

Data can be lost across an interface, one module can have adverse effect on another sub function when combined it may not produce the desired major functions. Integration testing is a systematic testing for constructing test to uncover errors associated within an interface.

The objectives taken from unit tested modules and a program structure is built for integrated testing.

All the modules are combined and the test is made.

A correction made in this testing is difficult because the vast expenses of the entire program complicated the isolation of causes. In this integration testing step, all the errors are corrected for next testing process.

### Validation Testing

After the completion of the integrated testing, software is completely assembled as a package; interfacing error has been uncovered and corrected and a final series of software test validation begins.

Validation testing can be defined in many ways but a simple definition is that validation succeeds when the software function in a manner that can be reasonably expected by the customer. After validation test has been conducted, one of two possible conditions exists:

### Output Testing

The next process of validation testing, is output testing of the proposed system, since no system could be successful if it does not produce the required output in the specified format. Asking the user about the format required, list the output to be generated or displayed by the system under considerations.

Output testing is a different test whose primary purpose is to fully exercise the computer based system although each test has a different purpose all the work should verify that all system elements have been properly integrated and perform allocated functions.

The output format on the screen is found to be corrected as the format was designed in the system design phase according to the user needs for the hard copy also; the output testing has not resulted in any correction in the system.

**IMPLEMENTATION**

System implementation is the stage of the project that the theoretical design is turned into a working system. If the implementation stage is not properly planned and controlled, it can cause error. Thus it can be considered to be the most crucial stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective.

Normally this stage involves setting up a coordinating committee, which will act as a sounding board for ideas; complaints and problem. The first task is implementation planning; i.e., deciding on the methods and time scale to be adopted. Apart from planning two major task of preparing for implementation are, education takes place much earlier in the project; at the implementation stage the emphasis must be on training in new skills to give staff confidence they can use the system. Once staff has been trained, the system can be tested.

After the implementation phase is completed and the user staff is adjusted to the changes created by the candidate system, evaluation and maintenance is to bring the new system to standards.

# CONCLUSION

In conclusion, fuel automation systems, or automatic tank gauging systems, offer several advantages to fuel retailers. These systems provide accurate and real-time fuel level information, improve safety, and reduce labor costs. They also help fuel retailers manage their inventory more efficiently and comply with regulations. While fuel automation systems do have some drawbacks, such as high initial costs and maintenance requirements, their benefits generally outweigh the drawbacks, making them a worthwhile investment for many fuel retailers. By implementing fuel automation systems, fuel retailers can improve their operations, reduce risks, and increase profitability.

.

## BIBLIOGRAPY

**Books Referred:**

* Naftalin, Maurice, and Philip Wadler. Java Generics and Collections. O'Reilly Media, 2006.
* Shildt, Herbert. Java: A Beginner's Guide. McGraw-Hill Education, 2018.
* Sierra, Kathy, and Bert Bates. Head First Java, 2nd Edition. O'Reilly Media, 2005.
* Subramaniam, Venkat. Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions. Pragmatic Bookshelf, 2014.

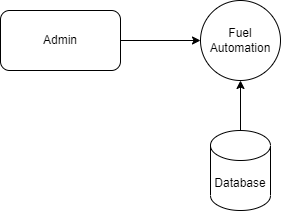
## Websites:

* Vogella. "Java Tutorials." Vogella, 2023, <https://www.vogella.com/tutorials/java.html>.
* GitHub. "Java Repositories." GitHub, <https://github.com/topics/java>.
* JavaWorld. "Java Tutorials and Articles." JavaWorld, https://www.javaworld.com/.
* JetBrains. "IntelliJ IDEA Documentation." JetBrains, <https://www.jetbrains.com/help/idea/>.
* Spring Framework. "Spring Boot Documentation." Spring Framework, https://docs.spring.io/spring-boot/docs/current/reference/html/.

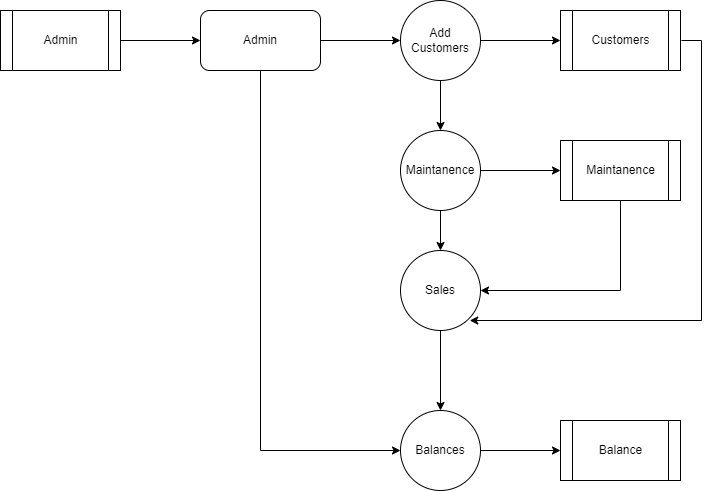
# APPENDICES

## Data Flow Diagram

**Level 0**



**Level 1**



## TABLE STRUCTURE

**Table Name :** Customer

**Primary Key :** Customer\_id

**Table Description :** This table is used to maintain the details about customer

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Customer\_id | Int | 8 | Primary Key |
| Customer code | Varchar | 15 | Not null |
| Full name | Varchar | 10 | Not null |
| Contact | Int | 10 | Not null |
| Email | Varchar | 26 | Not null |
| Address | Varchar | 50 | Not null |

**Table Name :** Payment

**Primary Key :** Payment\_id

**Table Description :** This table is used to maintain the details about payment

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Payment\_id | Int | 8 | Primary Key |
| Customer\_id | Int | 10 | Foreign key |
| Amount | Int | 10 | Not null |
| Date | Date | 10 | Not null |

**Table Name :** Transaction

**Primary Key :** Transaction\_id

**Table Description :** This table is used to maintain the details about transaction

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Transaction\_id | Int | 8 | Primary Key |
| Customer\_id | Int | 10 | Foreign key |
| Reception\_no | Varchar | 10 | Not null |
| Petrol Type | Varchar | 10 | Not null |
| Price | Int | 10 | Not null |
| Liter | Int | 10 | Not null |
| Amount | Int | 10 | Not null |
| Discount | Int | 10 | Not null |
| Total | Int | 10 | Not null |
| User\_id | Int | 10 | Foreign key |

## B. Sample Coding

. <?java

session\_start();

require\_once('DBConnection.java');

Class Actions extends DBConnection{

function \_\_construct(){

parent::\_\_construct();

}

function \_\_destruct(){

parent::\_\_destruct();

}

function login(){

extract($\_POST);

$sql = "SELECT \* FROM user\_list where username = '{$username}' and `password` = '".md5($password)."' ";

@$qry = $this->query($sql)->fetchArray();

if(!$qry){

$resp['status'] = "failed";

$resp['msg'] = "Invalid username or password.";

}else{

$resp['status'] = "success";

$resp['msg'] = "Login successfully.";

foreach($qry as $k => $v){

if(!is\_numeric($k))

$\_SESSION[$k] = $v;

}

}

return json\_encode($resp);

}

function logout(){

session\_destroy();

header("location:./");

}

function save\_user(){

extract($\_POST);

$data = "";

foreach($\_POST as $k => $v){

if(!in\_array($k,array('id'))){

if(!empty($id)){

if(!empty($data)) $data .= ",";

$data .= " `{$k}` = '{$v}' ";

}else{

$cols[] = $k;

$values[] = "'{$v}'";

}

}

}

if(empty($id)){

$cols[] = 'password';

$values[] = "'".md5($username)."'";

}

if(isset($cols) && isset($values)){

$data = "(".implode(',',$cols).") VALUES (".implode(',',$values).")";

}

@$check= $this->query("SELECT count(user\_id) as `count` FROM user\_list where `username` = '{$username}' ".($id > 0 ? " and user\_id != '{$id}' " : ""))->fetchArray()['count'];

if(@$check> 0){

$resp['status'] = 'failed';

$resp['msg'] = "Username already exists.";

}else{

if(empty($id)){

$sql = "INSERT INTO `user\_list` {$data}";

}else{

$sql = "UPDATE `user\_list` set {$data} where user\_id = '{$id}'";

}

@$save = $this->query($sql);

if($save){

$resp['status'] = 'success';

if(empty($id))

$resp['msg'] = 'New User successfully saved.';

else

$resp['msg'] = 'User Details successfully updated.';

}else{

$resp['status'] = 'failed';

$resp['msg'] = 'Saving User Details Failed. Error: '.$this->lastErrorMsg();

$resp['sql'] =$sql;

}

}

return json\_encode($resp);

}

function delete\_user(){

extract($\_POST);

@$delete = $this->query("DELETE FROM `user\_list` where rowid = '{$id}'");

if($delete){

$resp['status']='success';

$\_SESSION['flashdata']['type'] = 'success';

$\_SESSION['flashdata']['msg'] = 'User successfully deleted.';

}else{

$resp['status']='failed';

$resp['error']=$this->lastErrorMsg();

}

return json\_encode($resp);

}

function update\_credentials(){

extract($\_POST);

$data = "";

foreach($\_POST as $k => $v){

if(!in\_array($k,array('id','old\_password')) && !empty($v)){

if(!empty($data)) $data .= ",";

if($k == 'password') $v = md5($v);

$data .= " `{$k}` = '{$v}' ";

}

}

if(!empty($password) && md5($old\_password) != $\_SESSION['password']){

$resp['status'] = 'failed';

$resp['msg'] = "Old password is incorrect.";

}else{

$sql = "UPDATE `user\_list` set {$data} where user\_id = '{$\_SESSION['user\_id']}'";

@$save = $this->query($sql);

if($save){

$resp['status'] = 'success';

$\_SESSION['flashdata']['type'] = 'success';

$\_SESSION['flashdata']['msg'] = 'Credential successfully updated.';

foreach($\_POST as $k => $v){

if(!in\_array($k,array('id','old\_password')) && !empty($v)){

if(!empty($data)) $data .= ",";

if($k == 'password') $v = md5($v);

$\_SESSION[$k] = $v;

}

}

}else{

$resp['status'] = 'failed';

$resp['msg'] = 'Updating Credentials Failed. Error: '.$this->lastErrorMsg();

$resp['sql'] =$sql;

}

}

return json\_encode($resp);

}

function save\_petrol\_type(){

extract($\_POST);

$data = "";

foreach($\_POST as $k => $v){

if(!in\_array($k,array('id'))){

$v = addslashes(trim($v));

if(empty($id)){

$cols[] = "`{$k}`";

$vals[] = "'{$v}'";

}else{

if(!empty($data)) $data .= ", ";

$data .= " `{$k}` = '{$v}' ";

}

}

}

if(isset($cols) && isset($vals)){

$cols\_join = implode(",",$cols);

$vals\_join = implode(",",$vals);

}

if(empty($id)){

$sql = "INSERT INTO `petrol\_type\_list` ({$cols\_join}) VALUES ($vals\_join)";

}else{

$sql = "UPDATE `petrol\_type\_list` set {$data} where petrol\_type\_id = '{$id}'";

}

@$check= $this->query("SELECT COUNT(petrol\_type\_id) as count from `petrol\_type\_list` where `name` = '{$name}' ".($id > 0 ? " and petrol\_type\_id != '{$id}'" : ""))->fetchArray()['count'];

if(@$check> 0){

$resp['status'] ='failed';

$resp['msg'] = 'Petrol Type already exists.';

}else{

@$save = $this->query($sql);

if($save){

$resp['status']="success";

if(empty($id))

$resp['msg'] = "Petrol Type successfully saved.";

else

$resp['msg'] = "Petrol Type successfully updated.";

}else{

$resp['status']="failed";

if(empty($id))

$resp['msg'] = "Saving New Petrol Type Failed.";

else

$resp['msg'] = "Updating Petrol Type Failed.";

$resp['error']=$this->lastErrorMsg();

}

}

return json\_encode($resp);

}

function delete\_petrol\_type(){

extract($\_POST);

@$delete = $this->query("DELETE FROM `petrol\_type\_list` where petrol\_type\_id = '{$id}'");

if($delete){

$resp['status']='success';

$\_SESSION['flashdata']['type'] = 'success';

$\_SESSION['flashdata']['msg'] = 'Petrol Type successfully deleted.';

}else{

$resp['status']='failed';

$resp['error']=$this->lastErrorMsg();

}

return json\_encode($resp);

}

function save\_customer(){

extract($\_POST);

if(empty($id)){

$prefix = date('Ym');

$code = sprintf("%'.04d",1);

while(true){

$check = $this->query("SELECT count(customer\_id) FROM customer\_list where customer\_code = '".$prefix.'-'.$code."' ")->fetchArray()[0];

if($check > 0){

$code = sprintf("%'.04d",abs($code) + 1);

}else{

break;

}

}

$\_POST['customer\_code'] = $prefix.'-'.$code;

}

$data = "";

foreach($\_POST as $k => $v){

if(!in\_array($k,array('id'))){

$v = addslashes(trim($v));

if(empty($id)){

$cols[] = "`{$k}`";

$vals[] = "'{$v}'";

}else{

if(!empty($data)) $data .= ", ";

$data .= " `{$k}` = '{$v}' ";

}

}

}

if(isset($cols) && isset($vals)){

$cols\_join = implode(",",$cols);

$vals\_join = implode(",",$vals);

}

if(empty($id)){

$sql = "INSERT INTO `customer\_list` ({$cols\_join}) VALUES ($vals\_join)";

}else{

$sql = "UPDATE `customer\_list` set {$data} where customer\_id = '{$id}'";

}

@$save = $this->query($sql);

if($save){

$resp['status']="success";

if(empty($id))

$resp['msg'] = "Customer successfully saved.";

else

$resp['msg'] = "Customer successfully updated.";

}else{

$resp['status']="failed";

if(empty($id))

$resp['msg'] = "Saving New Customer Failed.";

else

$resp['msg'] = "Updating Customer Failed.";

$resp['error']=$this->lastErrorMsg();

}

return json\_encode($resp);

}

function delete\_customer(){

extract($\_POST);

@$delete = $this->query("DELETE FROM `customer\_list` where customer\_id = '{$id}'");

if($delete){

$resp['status']='success';

$\_SESSION['flashdata']['type'] = 'success';

$\_SESSION['flashdata']['msg'] = 'Customer successfully deleted.';

}else{

$resp['status']='failed';

$resp['error']=$this->lastErrorMsg();

}

return json\_encode($resp);

}

function save\_transaction(){

extract($\_POST);

$data = "";

$prefix = date('Ym');

$code = sprintf("%'.04d",1);

$i = 0;

while(true){

$i++;

$chk = $this->query("SELECT count(transaction\_id) `count` FROM `transaction\_list` where receipt\_no = '".$prefix.'-'.$code."' ")->fetchArray()['count'];

if($chk > 0){

$code = sprintf("%'.04d",abs($code) + 1);

}else{

break;

}

}

$\_POST['receipt\_no'] = $prefix.'-'.$code;

$\_POST['user\_id'] = $\_SESSION['user\_id'];

foreach($\_POST as $k => $v){

if(!in\_array($k,array('id')) && !is\_array($\_POST[$k])){

$v = addslashes(trim($v));

if(empty($id)){

$cols[] = "`{$k}`";

$vals[] = "'{$v}'";

}else{

if(!empty($data)) $data .= ", ";

$data .= " `{$k}` = '{$v}' ";

}

}

}

if(isset($cols) && isset($vals)){

$cols\_join = implode(",",$cols);

$vals\_join = implode(",",$vals);

}

if(empty($id)){

$sql = "INSERT INTO `transaction\_list` ({$cols\_join}) VALUES ($vals\_join)";

}else{

$sql = "UPDATE `transaction\_list` set {$data} where stock\_id = '{$id}'";

}

@$save = $this->query($sql);

if($save){

$resp['status']="success";

$\_SESSION['flashdata']['type']="success";

if(empty($id))

$\_SESSION['flashdata']['msg'] = "Transaction successfully saved.";

else

$\_SESSION['flashdata']['msg'] = "Transaction successfully updated.";

if(empty($id))

$last\_id = $this->query("SELECT last\_insert\_rowid()")->fetchArray()[0];

$tid = empty($id) ? $last\_id : $id;

if($type == 2){

$debt\_data = "('{$tid}','{$customer\_id}','{$total}')";

$save\_debts = $this->query("INSERT INTO `debt\_list` (`transaction\_id`,`customer\_id`,`amount`) VALUES {$debt\_data}");

}

$resp['transaction\_id'] = $tid;

}else{

$resp['status']="failed";

if(empty($id))

$resp['msg'] = "Saving New Transaction Failed.";

else

$resp['msg'] = "Updating Transaction Failed.";

$resp['error']=$this->lastErrorMsg();

}

return json\_encode($resp);

}

function delete\_transaction(){

extract($\_POST);

@$delete = $this->query("DELETE FROM `transaction\_list` where transaction\_id = '{$id}'");

if($delete){

$resp['status']='success';

$\_SESSION['flashdata']['type'] = 'success';

$\_SESSION['flashdata']['msg'] = 'Transaction successfully deleted.';

}else{

$resp['status']='failed';

$resp['error']=$this->lastErrorMsg();

}

return json\_encode($resp);

}

function save\_payment(){

extract($\_POST);

if(empty($id)){

$prefix = date('Ym');

$code = sprintf("%'.04d",1);

while(true){

$check = $this->query("SELECT count(payment\_id) FROM payment\_list where payment\_code = '".$prefix.'-'.$code."' ")->fetchArray()[0];

if($check > 0){

$code = sprintf("%'.04d",abs($code) + 1);

}else{

break;

}

}

$\_POST['payment\_code'] = $prefix.'-'.$code;

}

$data = "";

foreach($\_POST as $k => $v){

if(!in\_array($k,array('id'))){

$v = addslashes(trim($v));

if(empty($id)){

$cols[] = "`{$k}`";

$vals[] = "'{$v}'";

}else{

if(!empty($data)) $data .= ", ";

$data .= " `{$k}` = '{$v}' ";

}

}

}

if(isset($cols) && isset($vals)){

$cols\_join = implode(",",$cols);

$vals\_join = implode(",",$vals);

}

if(empty($id)){

$sql = "INSERT INTO `payment\_list` ({$cols\_join}) VALUES ($vals\_join)";

}else{

$sql = "UPDATE `payment\_list` set {$data} where payment\_id = '{$id}'";

}

@$save = $this->query($sql);

if($save){

$resp['status']="success";

if(empty($id))

$resp['msg'] = "Payment successfully added.";

else

$resp['msg'] = "Payment successfully updated.";

}else{

$resp['status']="failed";

if(empty($id))

$resp['msg'] = "Saving New Payment Failed.";

else

$resp['msg'] = "Updating Payment Failed.";

$resp['error']=$this->lastErrorMsg();

}

return json\_encode($resp);

}

}

$a = isset($\_GET['a']) ?$\_GET['a'] : '';

$action = new Actions();

switch($a){

case 'login':

echo $action->login();

break;

case 'customer\_login':

echo $action->customer\_login();

break;

case 'logout':

echo $action->logout();

break;

case 'customer\_logout':

echo $action->customer\_logout();

break;

case 'save\_user':

echo $action->save\_user();

break;

case 'delete\_user':

echo $action->delete\_user();

break;

case 'update\_credentials':

echo $action->update\_credentials();

break;

case 'save\_petrol\_type':

echo $action->save\_petrol\_type();

break;

case 'delete\_petrol\_type':

echo $action->delete\_petrol\_type();

break;

case 'save\_customer':

echo $action->save\_customer();

break;

case 'delete\_customer':

echo $action->delete\_customer();

break;

case 'save\_transaction':

echo $action->save\_transaction();

break;

case 'delete\_transaction':

echo $action->delete\_transaction();

break;

case 'save\_payment':

echo $action->save\_payment();

break;

default:

// default action here

break;

}

## D. Sample Input

## Admin login page

## 

## Input of new customer page

## 

## Input of payment page

## 

## Input of Transaction

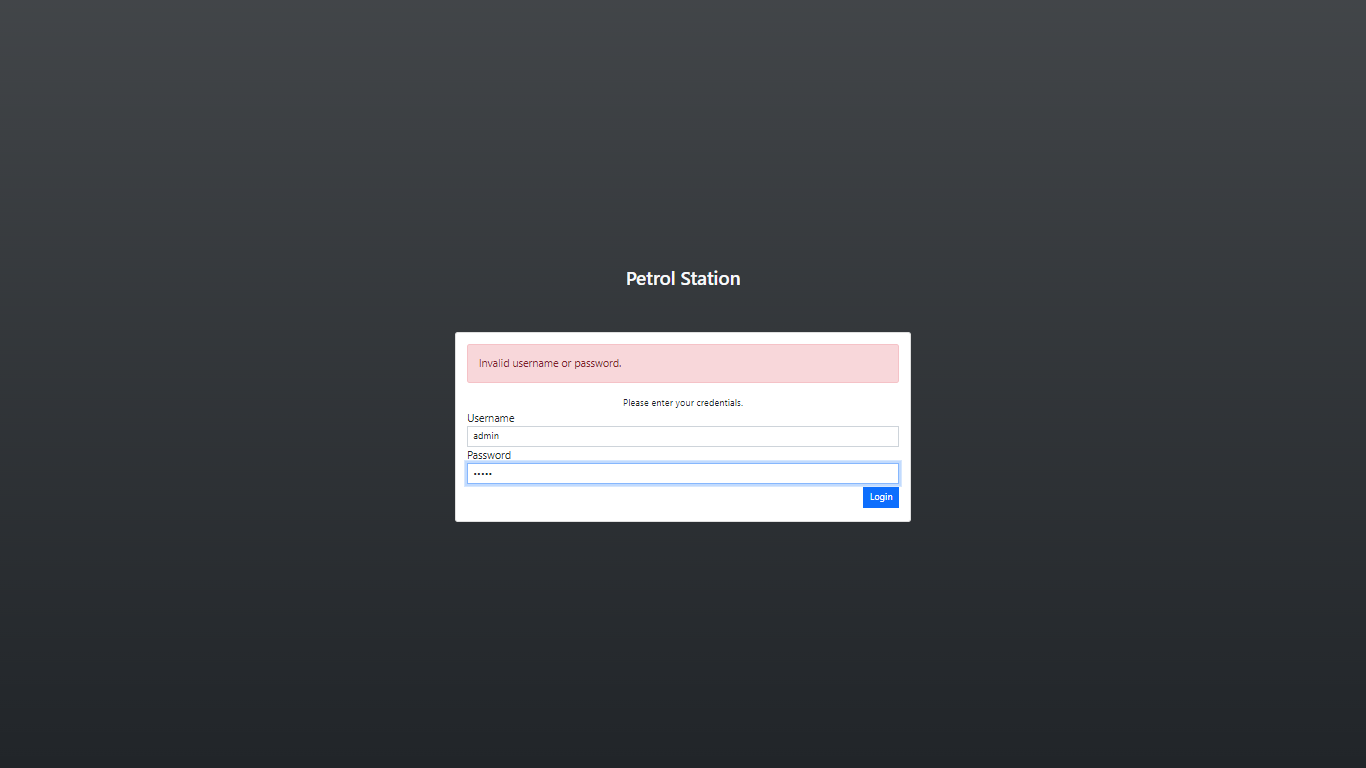
## 

## Input of new user

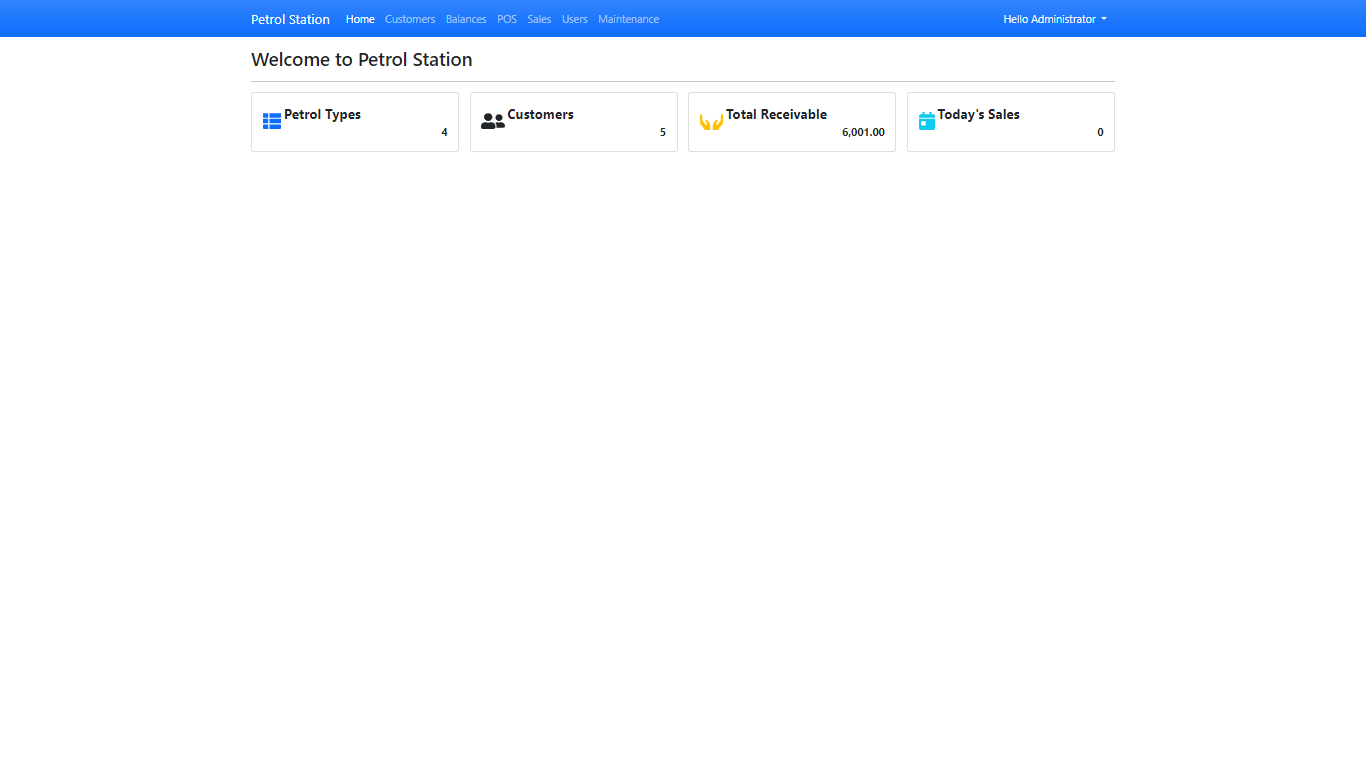
## 

## E. Sample Output

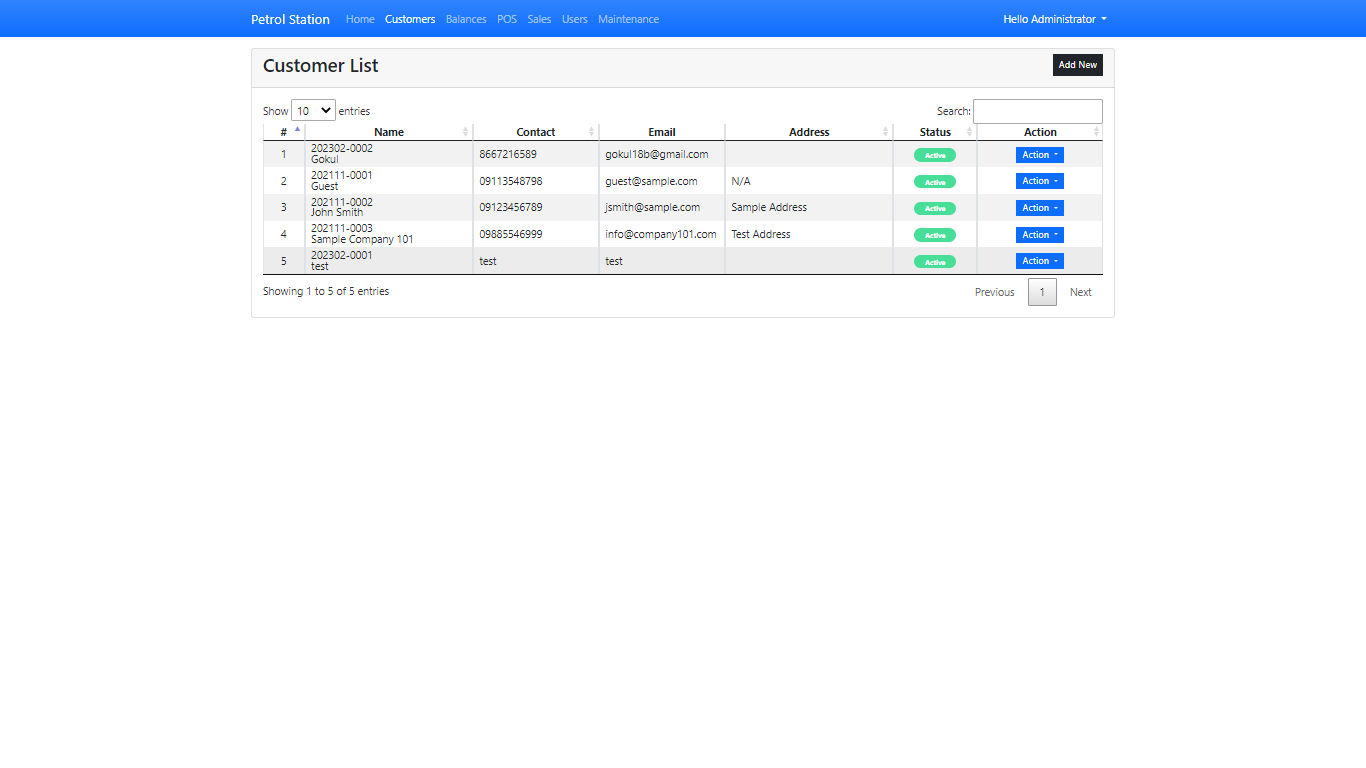
**Output of Invalid username or password**



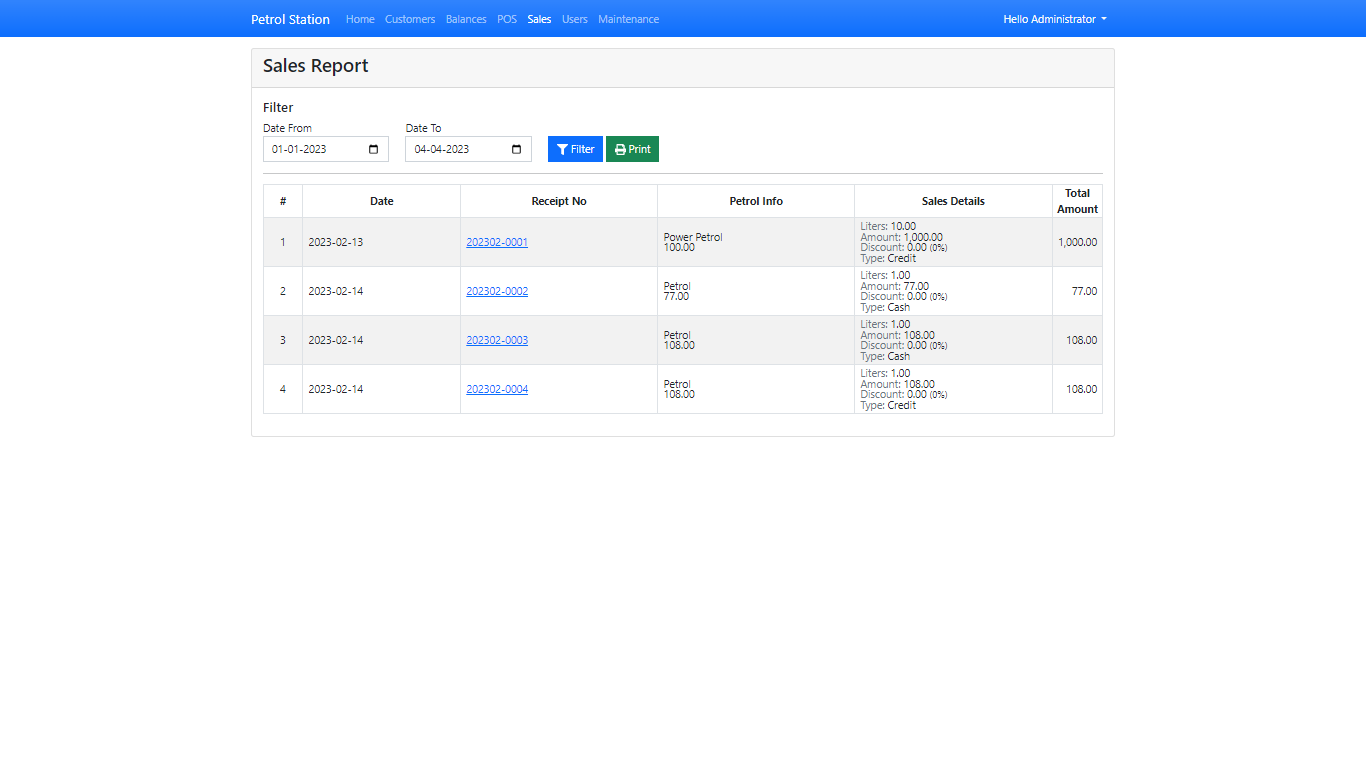
Output of Dashboard



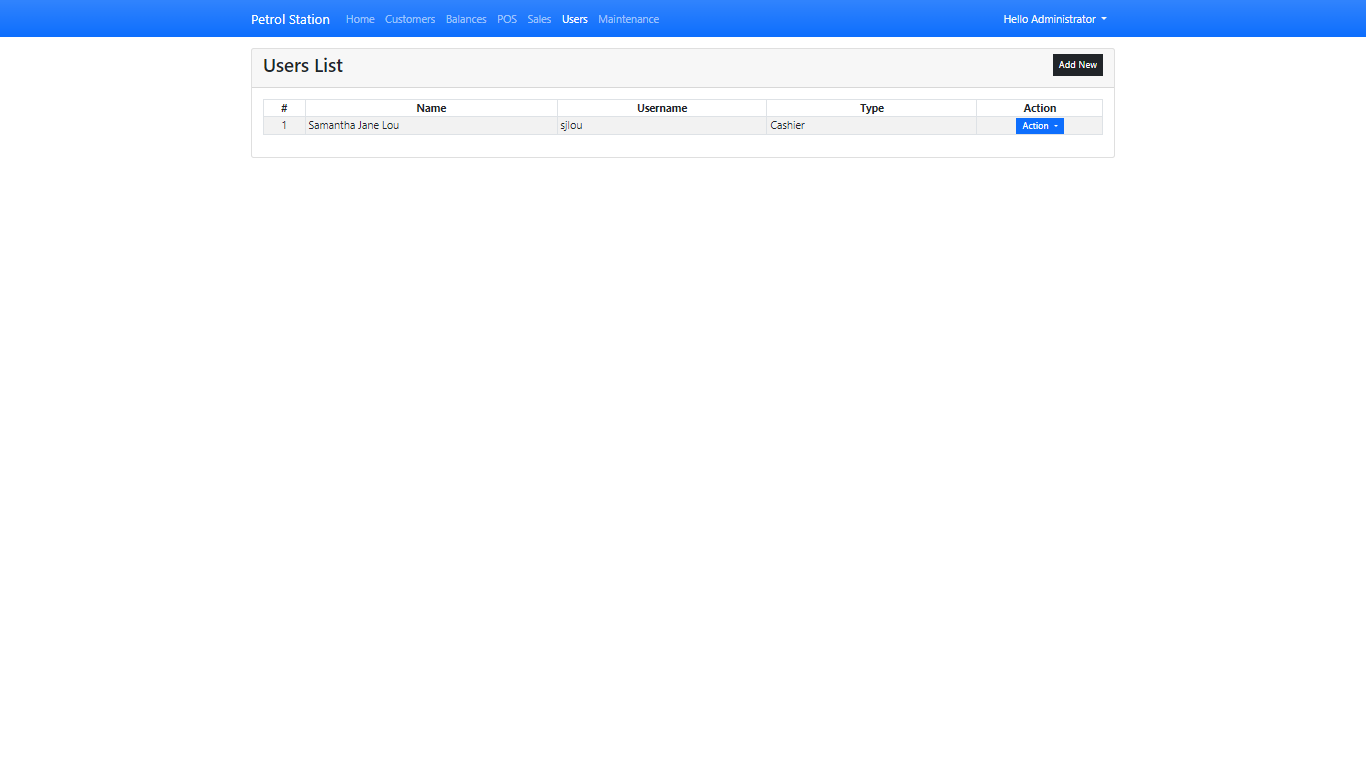
Output of Customer Details



Output of Sales Report



Output of User list



Output of Maintenance

