Fuel Delivery on Demand

Mini project report submitted to

St. Berchman’s Autonomous

College,

Changanacherry

In partial fulfilment of the requirement for the award of the BACHELOR OF COMPUTER APPLICATION

BY

JOHN THOMAS PUTHEN 12012578

JOSEPH ANTONY 12012579

 Under the guidance of Mrs. Dhanya C Nair

DEPARTMENT OF COMPUTER SCIENCE ST. BERCHMAN’S AUTONOMOUS COLLEGE,

CHANGANACHERRY

DECEMBER, 2022

ST BERCHMAN’S AUTONOMOUS COLLEGE, CHANGANACHERRY

DEPARTMENT OF COMPUTER SCIENCE



**CERTIFICATE**

**Certify that the report entitled “*FUELWAY”* is a bonafide record of the project work done by *John Thomas Puthen* (12012578) and Joseph Antony(12012579) under our guidance and supervision is submitted in partial fulfilment of the Bachelor Degree in Computer Applications, awarded by Mahatma Gandhi University Kerala and that no part of this work has been submitted earlier for the award of any other degree.**

## Project guide HOD

**Submitted for Viva Voce on …………………………………**

## External Examiner

**1.**

**2.**

# DECLARATION

I hereby declare that this project work entitled “FUELWAY” submitted to Mahatma Gandhi University in partial fulfilment of the requirement for the award of the degree of Bachelor of Computer Application is a record of original work done by me under the Guidance of Mrs. Dhanya C Nair, during my period of project work in St. Berchmans College, Changanacherry.

JOHN THOMAS PUTHEN JOSEPH ANTONY

# ACKNOWLEDGEMENT

Firstly, I would like to thank Almighty for his abundant grace upon me.

I express my heartfelt thanks to our principle, Rev. Reji P Kurian for his warm support with regard to my work and to The College Management for providing the facilities required.

I would like to place my deep sense of gratitude to Ms. Dhanya C Nair (Head of the Department, Dept of Computer Science, S.B College, Changanacherry) for her invaluable help and guidance out the project work to a success.

I express my sincere gratitude to Ms. Dhanya C Nair (Department of computer science, S.B College Changanacherry) for her valuable help, assessment and guidance.

I also extend my sincere thanks to all other members of the faculty of the department of Computer Science for their assistance and encouragement throughout the project.

Finally I would like to thank my friends and well-wishers for their co-ordination and encouragement.

JOHN THOMAS PUTHEN

JOSEPH ANTONY

# ABSTRACT

The **Fuel Delivery on Demand** application to develop delivery on-demand fuel depends on the user order and request. Due to growth of automobiles in the market, fuel consumption became more. In existing system, unfortunately because of some reason if vehicle stops due to lack of petrol, it will be very hard for the owner to push the vehicle to the nearest petrol pump. In some cases people go to new location and sometimes they won’t be having any idea of the gas stations to refuel their vehicles. The proposed system to develop application to deliver the fuel to those who need to refuel vehicles at any location and time.

In this application three modules using user, fuel station, admin. Admin can verify Fuel Station details, then it will see user modules. Fuel Station can add their branch information like address and google map geo location and kind of fuel provide. User can search by locality or station name and book the fuel on online app. Our objective develop using angular java script and MySQL as our backend database with responsive applicaion.

**CONTENTS Page Number**

1. INTRODUTION 1
2. SOFTWARE REQUIREMENT SPECIFICATION 2
   1. SYSTEM REQUIREMNT 2
3. [SYSTEM ANALYSIS 3](#_TOC_250008)
   1. [EXISTING SYSTEM 4](#_TOC_250007)
   2. [PROPOSED SYSTEM 5](#_TOC_250006)
   3. MODLES 6
4. [SYSTEM DESIGN 7](#_TOC_250005)
   1. [INPUT DESIGN 8](#_TOC_250004)
   2. [DATA FLOW DIAGRAM 9](#_TOC_250003)
   3. DATA BASE DESIGN 10

4.5OUTPUT DESIGN 11

1. [SYSTEM TESTING AND IMPLEMENTATION 12](#_TOC_250002)
   1. [TYPES OF TESTING 13](#_TOC_250001)
2. [FUTURE ENHANCEMENT 14](#_TOC_250000)
3. FORM LAYOUT 15
4. SAMPLE CODE 16
5. CONCLUSION 17

# INTRODUCTION

**Fuelway is an online platform to provide doorstep delivery of fuel in a safe, reliable, and efficient way. You can place your order online and get fuel delivered to your location in specialized Refuellers, according to your chosen schedule.**

**Fuelway sources fuel from authorized and selected dealers of oil companies near your location. We follow a strict quality assurance program to ensure every drop of fuel is checked for quality before it gets loaded into our Refuellers for delivery. We do not purchase or stock fuel prior to delivery. Fuel is sourced only upon receiving orders from customers. We are making refueling smarter!**

There are 2 modules Admin & customer

**Admin:** The web-based admin panel should give administrators a holistic view of the company’s operations.

**Customer:** Customer can login by using their registered user id and password. They can order the amount of fuel they need within the location.

# SOFTWARE REQUIREMENT SPECIFICATION

A software requirement specification (SRS) is a description of a software system to be developed, laying out functional and non-functional requirements (Non-functional requirements impose constraints on the design or implementation such as performance engineering requirements, quality standards, or design constraints). The specification may include a set of use cases that describe interactions the user will have with the software.

Software requirement specification establishes the basis for agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by marketing and development division) on what the software product is to do as well as what it is not expected to do. Software requirement specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks and schedules.

The software requirement specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

# SYSTEM REQUIREMENTS

## Hardware requirements:

Hardware is a set of physical components, which performs the functions of applying appropriate, predefined instructions. In other words, one can say that electronic and mechanical parts of computers constitute hardware.

This project is built on a powerful programming language - PHP. It is a powerful server-side scripting language. The backend is SQL Server, which is used to maintain database and the GUI is designed by using web technologies like html, CSS and JavaScript. It can run on almost all the popular microcomputers.

## Software requirements:

The software is a set of procedures of coded information or a program which when fed into the computer hardware enables the computer to perform the various tasks. Software is like a current inside wire, which cannot be seen but its effect can be felt.

## HARDWARE REQUIREMENTS

* PROCESSOR : Intel Pentium 4 /AMD Equivalent.
* RAM : 2GB
* HARD DISK DRIVE : At least 40GB
* KEYBOARD : Standard Multimedia Keyboard
* MOUSE : Standard 2 button Mouse
* MONITOR : Standard viewable color Monitor
* Internet Access
* Server OS

# SOFTWARE REQUIREMENTS

OPERATING SYSTEM : Windows, Linux or Mac FRONT END : PHP,CSS,JAVASCRIPT,HTML

BACK END : MYSQL

TOOLS : Text editor, WampServer/Xampp

**About PHP**

**Introduction to PHP:**

* The full form of PHP is “Hypertext Preprocessor”. Its original name was “Personal Home Page”.
* Rasmus Lerdorf software engineer, Apache team member is the creator and original driving force behind PHP. The first part of PHP was developed for his personal use in late 1994.
* By the middle of 1997, PHP was being used on approximately 50,000 sites worldwide.
* PHP is server-side scripting language, which can be embedded in HTML or used as a stand-alone.
* PHP doesn’t do anything about what a page looks and sounds like. In fact, most of what PHP does is invisible to the end user.
* Someone looking at a PHP page will not necessarily be able to tell that it was not

written purely in HTML, because usually the result of PHP is HTML.

* PHP is an official module of Apache HTTP Server.
* PHP is fully cross-platform, meaning it runs native on several flavors of Unix, as well as on Windows and now on Mac OS X.

## Advantages of PHP

* Cost: PHP costs you nothing. It is open source software and doesn’t need to purchase it for development.
* Ease of Use: PHP is easy to learn, compared to the others. A lot of Ready-made PHP scripts are freely available in market so, you can use them in your project or get some help from them.
* HTML- Support: PHP is embedded within HTML; In other words, PHP pages are ordinary HTML pages that escape into PHP mode only when necessary. When a client requests this page, the web server preprocesses it. This means it goes through the page from top to bottom, looking for sections of PHP, which it will try to resolve.
* Cross-platform compatibility: MySQL run native on every popular flavor of Unix and windows. A huge percentage PHP and of the world’s HTTP servers run on one of these two classes of operating system.
* PHP is compatible with the three leading Web servers: Apache HTTP Server for Unix and Windows, Microsoft Internet Information Server, and Netscape Enterprise Server. It also works with several lesser-known servers, including Alex Blits’ fhttpd, Microsoft’s Personal Web Server, AOL Server and Omnicentrix’s Omni server application server.
* Stability: The word stable means two different things in this context:
* The server doesn’t need to be rebooted often
* The software doesn’t change radically and incompatiblyfrom release to release.
* To our advantage, both of these apply to both MySQL and PHP.Speed: PHP is pleasingly zippy in its execution, especially when compiled as and Apache module on the Unix side. Although it takes a slight performance 2hit by being interpreted

rather than compiled, this is far outweighed by the benefits PHP drives from its status as a Web server module.

## About MySQL

* MySQL Database Management System
* MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by MySQL AB. It is a commercial company, founded by the MySQL developers. It is a second generation Open Source Company that unites Open Source values and methodology with a successful business model.
* The MySQL Web site [(http://www.](http://www.mysql.com/))m[ysql.com/)](http://www.mysql.com/)) provides the latest information about MySQL software and MySQL AB.
* The official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”), but we don't mind if you pronounce it as “my sequel” or in some other localized way.
* MySQL is a database management system.
* MySQL is a relational database management system.
* MySQL software is Open Source.
* The MySQL Database Server is very fast, reliable, and easy to use.
* MySQL Server works in client/server or embedded systems.
* A large amount of contributed MySQL software is available**.**

**3. SYSTEM ANALYSIS**

The objectives of system analysis phase are establishment of requirements from system to be acquired, developed and installed. Analyzing the project to understand the intricacy form the vital part for system study. Problematic areas are identified and information is collected. Fact finding or gathering is essential to any analysis of requirement. It is necessary that the analyst familiarize himself with the objectives, activities, and functions of the organization in which the system is to be implemented.

**3.1 Existing system**

Everything is paper based hence it is time consuming.

In existing system there are lot of paper work and manual processing.

While writing a paper record the management have to keep the records very carefully as the entire data is written in those records.

## Proposed System

Now-a-days world is beneath computers and internet, the revolution is taking place very faster so the existing manual system should be changed to compete with other business people. Proposed system is developed to tackle all limitations of the existing system.

Objectives of the proposed system

* + - C Saves times for users
    - Manual work will be reduced
    - Quick and efficient
    - Produces various reports on deposits
    - Loss of data and redundancy is avoided
    - Paper work is reduced to computers

MODULE.

**Admin:** Admin login to his homepage with a specific username and password. He can add different book. He could view the complaint and reply them. Then he can add student/teacher.

**Customer:** Customer can login by using there registered user id and password. They can request for the book that is provided in the site. They can also report the book that they have read

**4. SYSTEM DESIGN**

Based on the user requirement and the detailed analysis of the existing, the new system must be designed. This is the phase of the system designing. It is the most crucial phase in the developments of a system. The logical system design arrived at as a result of the systems analysis is converted into physical system design. Normally, the design proceeds in two stages:

* Preliminary or general design
* Structured or detailed design

Preliminary or general design: in preliminary or general design, the features of the new system are specified. The costs of implementing these features and the benefits to be derived are estimated. If the project is still considered to be feasible, we move to detailed design stage.

Structured or detailed design: in the detailed design stage, computer oriented work begins in earnest. At this stage, the design of the system becomes more structured. Structured design is a blueprint of a computer system solution to a given problem having the Computer Application same components and inter-relationships among the same components as the original problem. Input, output, databases, forms, codification schemes and processing specifications are drawn up in detail. On the design stage, the programming language and the hardware and the software platform in which the new system will run are also decided.

## Input Design

Input design is the process of converting a user-oriented description of the inputs to a computer based business system into a programmer-oriented specification.

Inaccurate input data is the most common cause of data processing errors. If poor input design-particularly where operators must enter data from source documents- permits bad data to enter a computer system, the outputs produced are of little value. The input design process was initiated in the study phase where, as a part of the feasibility study.

* + Input data were found to be available for establishing and maintaining master and transaction files for creating output records.
  + The most suitable types of input media, for either off-line or on-line devices, were selected after a study of alternative data capture techniques.

In the design phase, as discussed earlier in this unit, the input design process was continued.

Specifically:

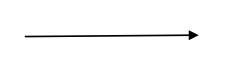
* + The decomposed data flow diagrams or expanded system flowchart identified master files (the database), transaction files, and the computer programs.
  + The input media selected in the study phase were reviewed. Additional studies of alternatives were performed as required, and tasks were allocated among equipment, manual operations, and computer programs.

## Data Flow Diagram

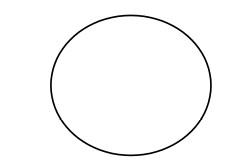
The data flow diagram is also known as “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design so it is the starting point of specification down to the lowest level of detail. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformation and the lines represent the data flow in the system.

DFD Symbols:

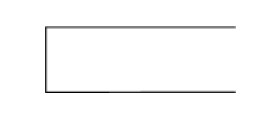
* + - A system defined a source or destination of data.
    - An arrow identifies data flow, data in motion.
    - A circle represents the process that transforms incoming data flow to outgoing data flow.
    - An open rectangular is data store-data at rest or a temporary repository of data.
* Arrow identifies Data Flow – Data in motion. It is a pipeline through which information flows. Data flow is a route, which enables packets travels from one point to another. Data may flow from a source to a processor and from data store or process. An arrow line depicts the flow, with arrow head pointing in the direction of the flow.



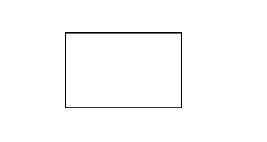
* A process represents a transformation where incoming data flows are changed into outgoing data flows.



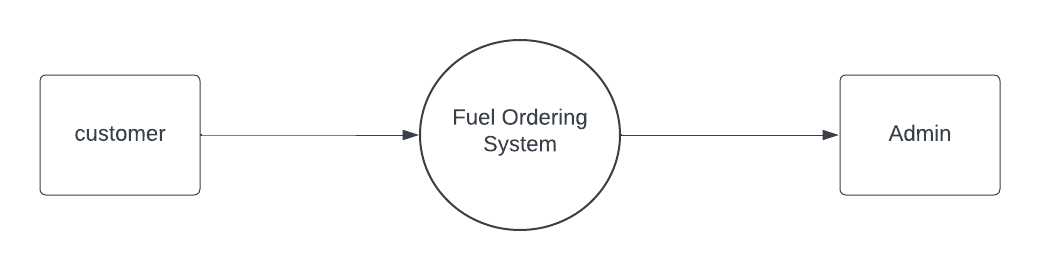
* A data store is a respiratory of data that is to be stored for use by one or more process may be as simple as buffer or queue or sophisticated as relational database. They should have clear names. If a process merely uses the content of store and does not alter it, the overhead goes from the store to the process. If a process alters the details in the store then a headed arrow is used.



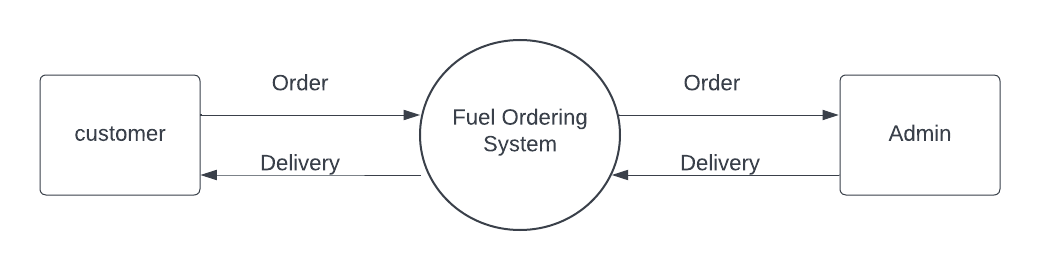
* A square defines a Source or Destination of system data. A source or sink is a person or part of an organization which enters or receives information from the system but is considered to be the contest of data flow model.



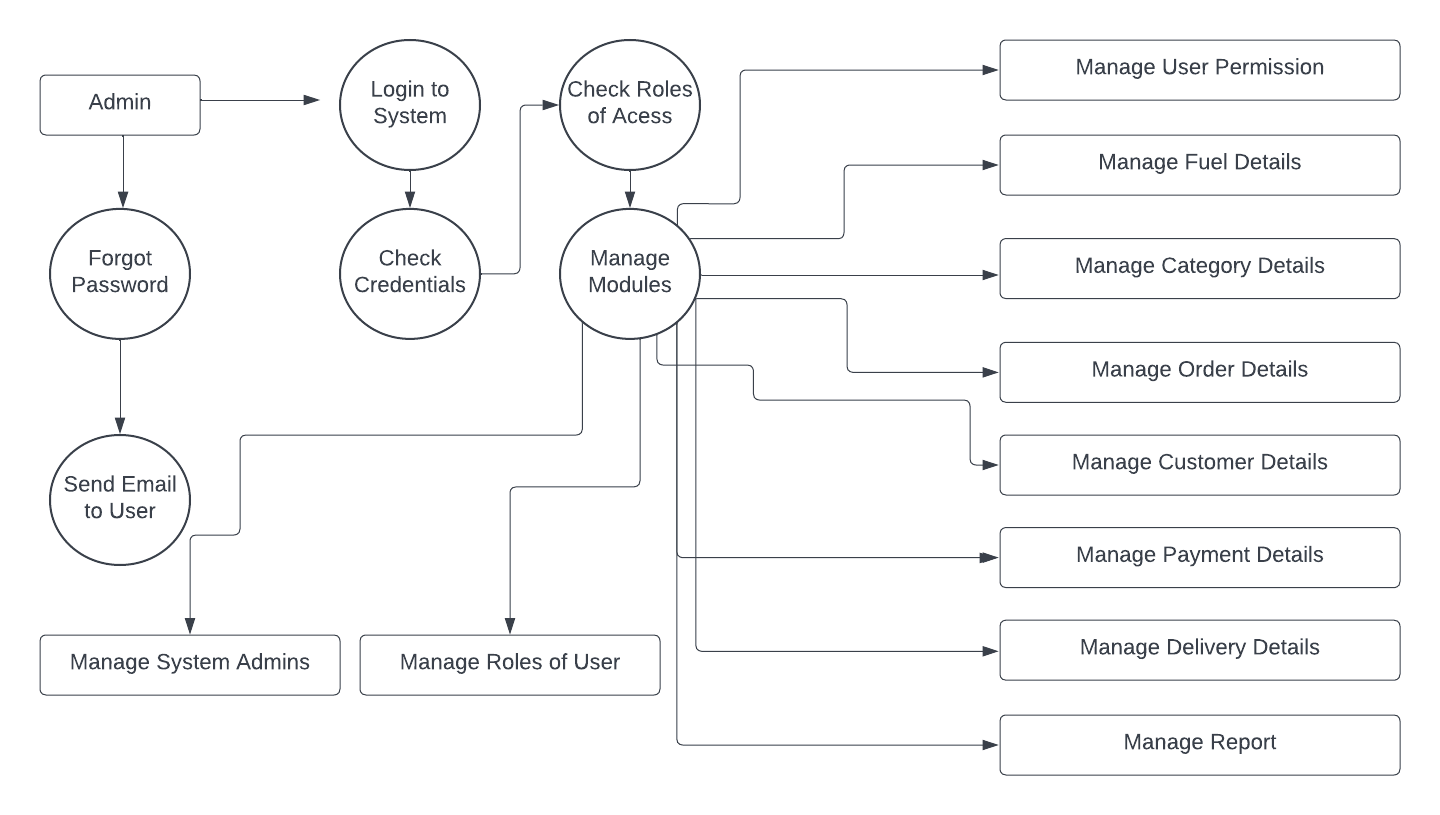
# LEVEL 0 DFD / Context Diagram

****

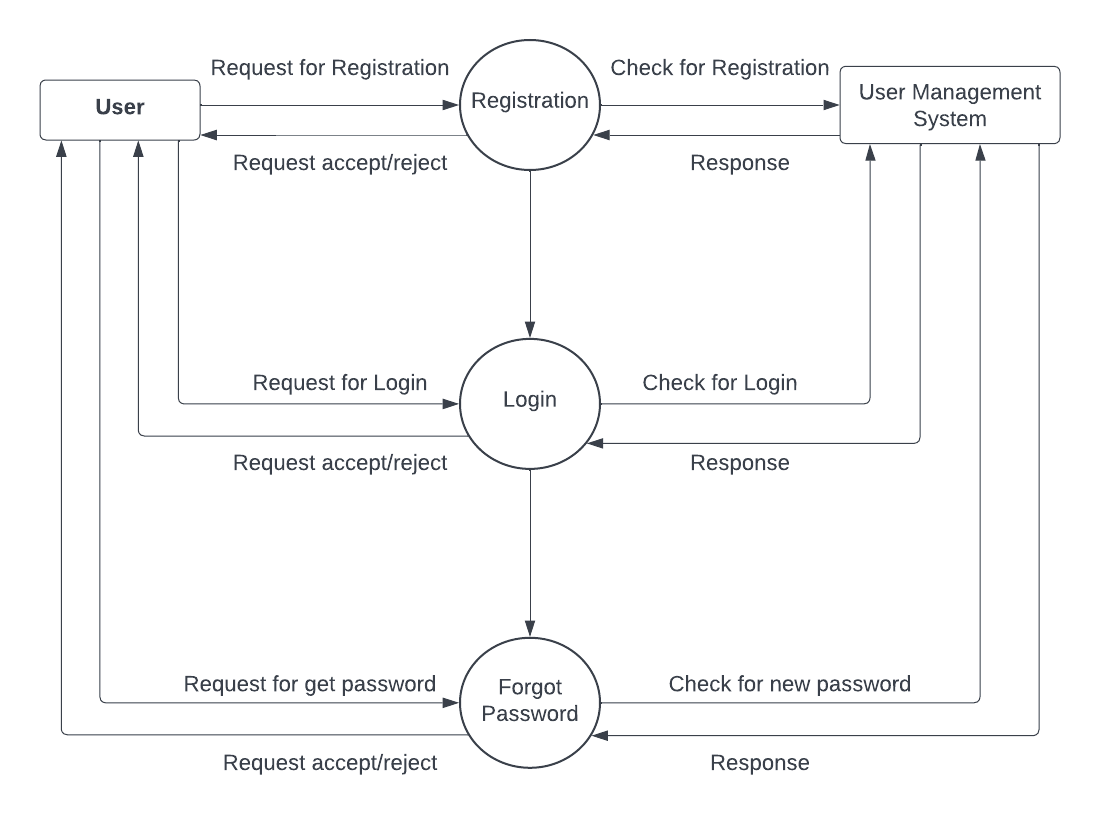
**LEVEL 0.1 DFD**

****

# ADMIN SIDE

****

**USER SIDE.**

****

# Database design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

* + - Determine the relationships between the different data elements.
    - Superimpose a logical structure upon the on the basis of these relationships.

In the field of relational database design, normalization is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics-insertion, update, and deletion anomalies-that could lead to a loss of data integrity. A standard piece of database design guidance is that the designer should create a fully normalized design; selective normalization can subsequently be performed, but only for performance reasons. However, some modeling disciplines, such as the dimensional modeling approach to data warehouse design, explicitly recommend non-normalized designs, i.e. designs that in large part do not adhere to 3NF. Normalization consists of normal forms that are:

First Normal Form Tables are said to be in first normal form when

* + - * The table has a primary key.
      * No single attribute (column) has multiple values.
      * The non-key attributes (columns) depend on the primary key.

Second Normal Form: Tables are said to be in second normal form when

* The tables meet criteria of for first normal form
* If the primary key is a composite of attributes (contains multiple columns), the non-key attributes (columns) must depend on the whole key.

Third Normal Form: Tables are said to be in third normal form when

* The tables meet the criteria for second normal form.
* Each non-key attribute in a row does not depend on the key column exclusive of other non-key columns are eliminated.

Boyce codd Normal Form: Tables are said to be in Boyce Codd normal form

* The tables must meet the criteria for third normal form
* For each functional dependency (X->Y), X should be a super key.

# TABLES

## ADMIN

|  |  |  |
| --- | --- | --- |
| **Field name** | **Datatype** | **Key constraint** |
| username | varchar(10) | Primary key |
| email | Varchar(20) |  |
| password | Varchar(15) |  |

**USER**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Datatype** | **Key constraint** |
| username | varchar(10) | Primary key |
| email | Varchar(20) |  |
| password | Varchar(15) |  |

## Output Design

Computer output is the most important and direct source of information to the administrator. Efficient, intelligible output design should improve the system’s relationships with the appraiser. A major form output, reports, is a hard copy from printer.

One of the most important features of an information system for users is the output it produces. Without quality output, the entire system may appear to be so unnecessary that users will avoid using it, possibly causing it to fail.

The term output applies to any information produced by an 9information system, whether printed or displayed. When analyst design computer output they

– Identify specific output that is needed to meet the information requirements.

* Select methods for presenting information.
* Create document, report or other formats that contain information produced by the system.

## Objectives of output

The output from an information system should accomplish one or more of the following objectives:

* Convey information about past activities, current status, or projection of the future.
* Signal important events, opportunities, problems or warnings. - Trigger an action. - Confirm an action.

## Types of output

Whether the output is a formatted report or a simple listing of the contents of a file, a computer process will produce the output. System output may be:

* A Document
* A Message Depending upon the circumstances and the contents, the output may be displayed or printed.

Output contents originate from the following sources:

* Retrieval from a data store. - Transmission from process or system activity.
* Directly from an input source.

## SYSTEM TESTING AND IMPLEMENTATION

Before actually implementing the new system into operation, a test run of the system is done for removing the bugs, if any. It is an important phase of a successful system. After codifying the whole programs of the system, a test plan should be developed and run on a given set of test data. The output of the test run should match the expected results. Sometimes, system testing is considered a part of implementation process.

## Types of testing

* + 1. **Unit Testing**

Unit testing is the practice of testing small pieces of code, typically individual functions,

alone and isolated. If your test uses some external resource, like the network or a database, it’s not a unit test.

Unit tests should be fairly simple to write. A unit tests should essentially just give the function that’s tested some inputs, and then check what the function outputs is correct. In practice this can vary, because if your code is poorly designed, writing unit tests can be difficult. Because of that, unit testing is the only testing method which also helps you write better code – Code that’s hard to unit test usually has poor design.

In a sense, unit testing is the backbone. You can use unit tests to help design your code and keep it as a safety net when doing changes, and the same methods you use for unit testing are also applicable to the other types of testing. All the other test types are also constructed from similar pieces as unit tests, they are just more complex and less precise.

Unit tests are also great for preventing regressions – bugs that occur repeatedly. Many times there’s been a particularly troublesome piece of code which just keeps breaking no matter how many times I fix it. By adding unit tests to check for those specific bugs, you can easily prevent situations like that. You can also use integration tests or functional tests for regression testing, but unit tests are much more useful because they are very specific, which makes it easy to pinpoint and then fix the problem.

When should you use unit testing? Ideally all the time, by applying test-driven development. A good set of unit tests do not only prevent bugs, but also improve your code design, and make sure you can later refactor your code without everything completely breaking apart.

## Integration Testing

As the name suggests, in integration testing the idea is to test how parts of the system work together – the integration of the parts. Integration tests are similar to unit tests, but there’s one big difference: while unit tests are isolated from other components, integration tests are not. For example, a unit test for database access code would not talk to a real database, but an integration test would.

Integration testing is mainly useful for situations where unit testing is not enough. Sometimes

you need to have tests to verify that two separate systems – like a database and your app – work together correctly, and that calls for an integration test. As a result, when validating integration test results, you could for example validate a database related test by querying the database to check the database state is correct.

Integration tests are often slower than unit tests because of the added complexity. They also might need some set up or configuration, such as the setting up of a test database. This makes writing and maintaining them harder than unit tests, so you should focus on unit tests unless you absolutely need an integration test.

You should have fewer integration tests than unit tests. You should mainly use them if you need to test two separate systems together, or if a piece of code is too complex to unit test. But in the latter case, I would recommend fixing the code so it’s easy to unit test instead.

## System Testing

After the system is put together, system testing is performed. Here the system is tested against requirements to see if all the requirements are met and the system performance as specified by the requirements.

System Testing (ST) is a black box testing technique performed to evaluate the complete system, the system’s compliance against specified requirements. In system testing, the functionalities of the system are tested from an end-to-end perspective.

System testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and non- functional testing

## Acceptance Testing

Finally, acceptance testing is performed to demonstrate to the client, on the real life data of the operations of the system.

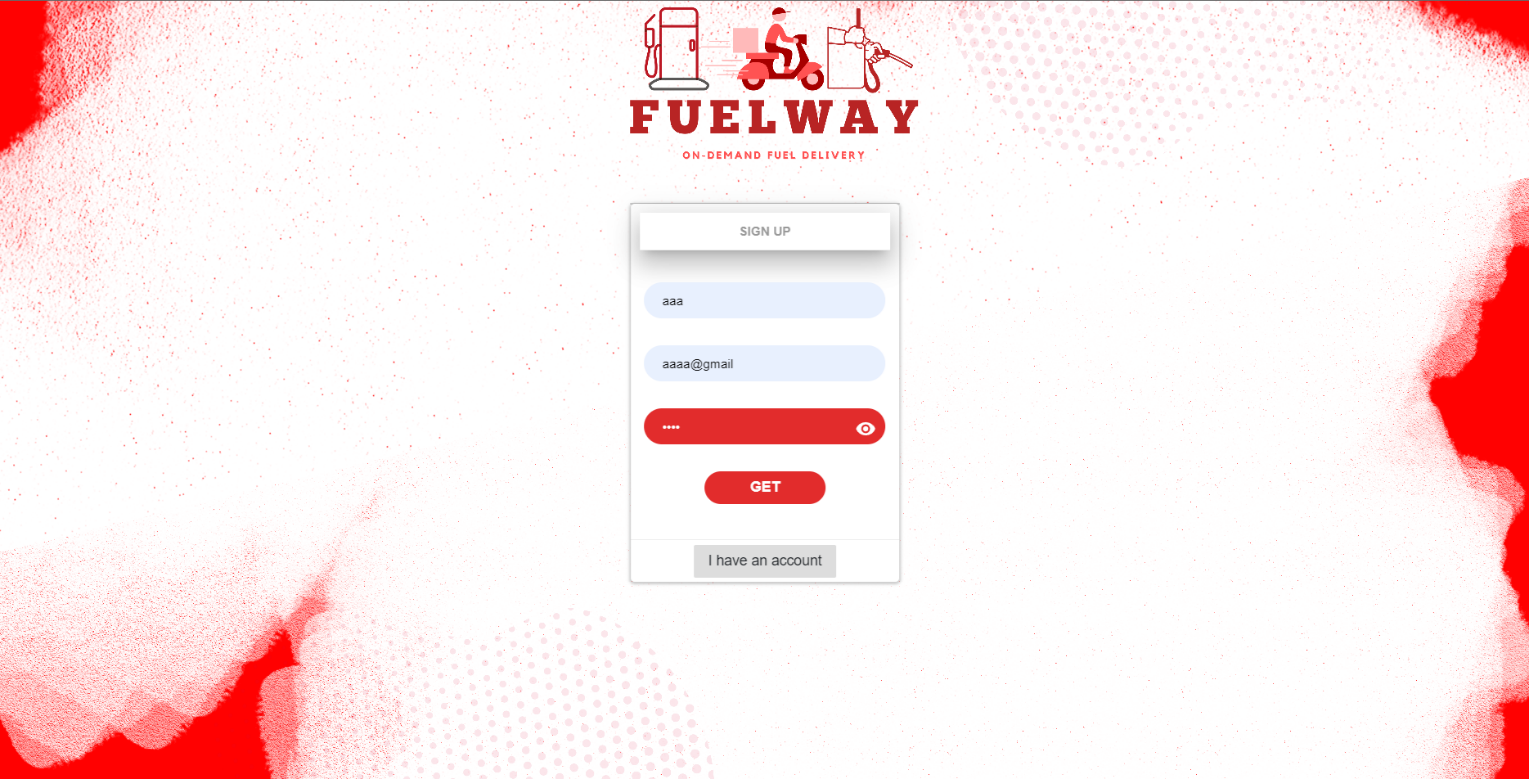
## FUTURE ENHANCEMENT

This system is designed in such a way that additional modification can be done in a very simple and efficient manner. Introducing the distributed database concept make further extensions to the system. The system can be changed very easily depending on changes in the organizational policies. The system is designed with the limited information available. So there are some drawbacks, but these drawbacks can easily be rectified by implementing additional techniques such as

1. Online system supporting.
2. Can use satellite communication system.
3. Providing additional security system.
4. Adaptive to any changes in government policy

# SYSTEM IMAGES

**ADMIN LOGIN**

****

**REGISTRATION / USER LOGIN**

****

# SAMPLE CODE

# LOGIN PAGE(HTML):

<html>

    <head>

        <!-- CSS only -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">

        <link rel="stylesheet" href="login.css">

        <body class="main-bg " background="for pass 01.png" style=" background-size: 100%; background-repeat: no-repeat; background-position: center center;background-size: cover;">

            <div class="main-wrapper">

                <div class="card-container">

                    <div class="card">

                        <div class="login-form">

                            <div class="header">Log in</div>

                            <div class="content">

                                <form method="post" action="login.php">

                                    <div class="input-field" >

                                        <input type="text" placeholder="Username" name="uname">

                                    </div>

                                    <div class="input-field">

                                        <input type="password" placeholder="Password" name="pass">

                                    </div>

                                    <div class="forget">

                                    <input type="checkbox" />Keep me Signed In

                                      <a href="forgot password.html">Forgot Password?</a>

                                    </div>

                                    <div class="input-field">

                                        <button class="btn btn-submit" type="submit">Log in</button>

                                    </div>

                                </form>

                            </div>

                            <div class="footer">

                                Don't have an account ?

                                <button class="btn btn-rotate" id="btn-signup">Sign up</button>

                            </div>

                        </div> <!-- end login-form panel -->

                        <div class="signup-form">

                            <div class="header">Sign up</div>

                            <div class="content">

                                <form action="register.php" method="post">

                                    <div class="input-field" >

                                        <input type="text" placeholder="Username" name="username">

                                    </div>

                                    <div class="input-field" >

                                        <input type="email" placeholder="Email" name="email">

                                    </div>

                                    <div class="input-field group">

                                        <input type="password" placeholder="Password" name="password">

                                        <span class="see-password">

                                            <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">

                                                <path d="M0 0h24v24H0z" fill="none"/>

                                                <path fill="#fff" d="M12 4.5C7 4.5 2.73 7.61 1 12c1.73 4.39 6 7.5 11 7.5s9.27-3.11 11-7.5c-1.73-4.39-6-7.5-11-7.5zM12 17c-2.76 0-5-2.24-5-5s2.24-5 5-5 5 2.24 5 5-2.24 5-5 5zm0-8c-1.66 0-3 1.34-3 3s1.34 3 3 3 3-1.34 3-3-1.34-3-3-3z"/>

                                            </svg>

                                        </span>

                                    </div>

                                    <div class="input-field">

                                        <button class="btn btn-submit" type="submit">Get started</button>

                                    </div>

                                </form>

                            </div>

                            <div class="footer">

                                <button class="btn btn-rotate" id="btn-login">I have an account</button>

                            </div>

                        </div> <!-- end signup-form panel -->

                    </div> <!-- end card -->

                </div> <!-- end card-container -->

            </div>

            <script src="login.js"></script>

    </body>

    </head>

</html>

**LOGIN PAGE (CSS):**

.main-wrapper {

    width: 100vw;

    overflow: hidden;

    min-height: 100vh;

    display: flex;

    justify-conent: center;

    align-items: center;

      font-size: 14px;

      font-family: "Helvetica Nueue",Arial,Verdana,sans-serif;

      /\*background:  url(https://github.com/Joseph89-creator/images/blob/main/fuelweay%20banner.png) no-repeat center;/\*

  background: #f87f7f;  /\* fallback for old browsers \*/

  background: -webkit-linear-gradient(to right, #909090,#909090);  /\* Chrome 10-25, Safari 5.1-6 \*/

  /\*background: linear-gradient(to right, #909090, #909090); \*//\* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ \*/

   z-index: 0;

  }

  /\* entire container, keeps perspective \*/

  .card-container {

      perspective: 500px;

      margin-bottom: 30px;

  }

.forget

{

    margin-left: 20px;

}

  /\* flip the pane when hovered \*/

  .card-container.rotate .card{

      transform: rotateY( 180deg );

  }

  /\* flip speed goes here \*/

  .card {

      transition: transform .85s;

      transform-style: preserve-3d;

      position: relative;

  }

  /\* hide back of pane during swap \*/

  .login-form, .signup-form {

      backface-visibility: hidden;

      position: absolute;

      top: 0;

      left: 0;

      background-color: #FFF;

      box-shadow: 0 1px 3px 2px rgba(0, 0, 0, 0.2);

      display: flex;

      flex-direction: column;

  }

  /\* front pane, placed above back \*/

  .login-form {

      z-index: 2;

  }

  /\* back, initially hidden pane \*/

  .signup-form {

      transform: rotateY(180deg);

      z-index: 3;

  }

  /\*        Style       \*/

  .card-container{

      width: 300px;

      margin: 0 auto;

  }

  .card{

      background: #FFFFFF;

      border-radius: 4px;

      color: #444444;

  }

  .card-container, .login-form, .signup-form {

      height: 420px;

      border-radius: 4px;

  }

  .login-form, .signup-form {

      width: 100%;

      height: 420px;

      border-radius: 4px;

  }

  .card .header {

      margin: 10px;

      padding: 10px 0 10px 0;

      text-align: center;

      border-bottom: 1px solid #EEEEEE;

      color: #999999;

      text-transform: uppercase;

      font-weight: bold;

      box-shadow: 0 16px 24px 2px rgba(0, 0, 0, 0.14), 0 6px 30px 5px rgba(0, 0, 0, 0.12), 0 8px 10px -7px rgba(0, 0, 0, 0.2);

  }

  .card .content{

      display: flex;

      align-items: center;

      justify-content: center;

      background-color: rgba(0, 0, 0, 0);

      box-shadow: none;

      flex: 1;

  }

  .card .footer {

      border-top: 1px solid #EEEEEE;

      color: #999999;

      padding: 5px 0 5px 0;

      text-align: center;

  }

  .btn {

      border: none;

      border-radius: 2px;

      display: inline-block;

      height: 36px;

      line-height: 36px;

      padding: 0 16px;

      font-family: inherit;

      font-weight: 100;

      text-align: center;

      vertical-align: middle;

      user-select: none;

      outline: none;

      border: none;

      cursor: pointer;

      transition: .4s;

  }

  .btn-rotate {

      color: #212529;

      background-color: #ddd;

  }

  .btn-rotate:hover {

      color: #212529;

      background-color: #ccc;

  }

  .btn-rotate:focus{

      outline: none;

      border: none;

  }

  .btn-submit {

      width: 50%;

      margin: 0 auto;

      border-radius: 20px;

      text-transform: uppercase;

      color: #ffffff;

      font-weight: bold;

      background-color: #e22c2c;

  }

  .btn-submit:hover {

      color: #ffffff;

      background-color: #e22c2c;

  }

  .btn-submit:focus{

      outline: none;

      border: none;

  }

  form{

      width: 100%;

  }

  .input-field{

      position: relative;

      width: 90%;

      margin: 20px auto;

      padding: 0;

      display: flex;

      align-items: center;

      justify-content: center;

  }

  .input-field input{

      width: 100%;

      max-height: 20px;

      padding: 20px;

      margin-bottom: 10px;

      border: none;

      outline: none;

      border-radius: 30px;

      background-color: #e22c2c;

      color: #fff;

      transition: .4s;

  }

  ::-webkit-input-placeholder {

    color: #fff;

  }

  ::-moz-placeholder {

    color: #fff;

  }

  :-ms-input-placeholder {

    color: #fff;

  }

  ::-ms-input-placeholder {

    color: #fff;

  }

  ::placeholder {

    color: #fff;

  }

  .input-field input:focus{

      box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.14), 0 3px 1px -2px rgba(0, 0, 0, 0.12), 0 1px 5px 0 rgba(0, 0, 0, 0.2);

  }

  .see-password{

      position: absolute;

      right: 10px;

      padding-bottom: 5px;

      cursor: pointer;

  }

  .group input{

      padding-right: 40px;

  }

**LOGIN PAGE (JS):**

document.addEventListener('DOMContentLoaded',  (event) => {

const rotateCard = () => {

  const cardContainer = document.querySelector('.card-container')

  cardContainer.classList.toggle('rotate')

}

const btnSignup = document.querySelector('#btn-signup') ,

      btnLogin = document.querySelector('#btn-login')

btnSignup.addEventListener('click', rotateCard)

btnLogin.addEventListener('click', rotateCard)

/\*See passwod\*/

const seePassword =  () => {

  const seePwdIcon = document.querySelector('.see-password'),

        pwdInput = document.querySelector('.group input')

  seePwdIcon.addEventListener('mousedown', () => {

    pwdInput.type = 'text'

  })

  seePwdIcon.addEventListener('mouseup', () => {

    pwdInput.type = 'password'

  })

  seePwdIcon.addEventListener('mouseover', () => {

    pwdInput.type = 'password'

  })

}

seePassword()

})

**LOGIN PAGE (PHP):**

<?php

session\_start();

include "connection.php";

if (isset($\_POST['uname']) && isset($\_POST['password'])) {

    function validate($data){

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

    }

    $uname = validate($\_POST['uname']);

    $pass = validate($\_POST['password']);

    if (empty($uname)) {

        header("Location: index.php?error=User Name is required");

        exit();

    }else if(empty($pass)){

        header("Location: index.php?error=Password is required");

        exit();

    }else{

        $sql = "SELECT \* FROM user01 WHERE username='$uname' AND password='$pass'";

        $result = mysqli\_query($conn, $sql);

        if (mysqli\_num\_rows($result) === 1) {

            $row = mysqli\_fetch\_assoc($result);

            if ($row['user\_name'] === $uname && $row['password'] === $pass) {

                echo "Logged in!";

                $\_SESSION['user\_name'] = $row['user\_name'];

                $\_SESSION['name'] = $row['name'];

                $\_SESSION['id'] = $row['id'];

                header("Location: home.php");

                exit();

            }else{

                header("Location: index.php?error=Incorect User name or password");

                exit();

            }

        }else{

            header("Location: index.php?error=Incorect User name or password");

            exit();

        }

    }

}else{

    header("Location: index.php");

    exit();

}

**REGISTRATION (PHP):**

<?php

    include($\_SERVER['DOCUMENT\_ROOT'].'/MINI/login/connection.php');

    $regusername = trim($\_POST['username']);

    $email = trim($\_POST['email']);

    $cmfpassword = trim($\_POST['password']);

            $sql = "INSERT INTO `user01`(`username`, `email`, `password`) VALUES ('$regusername','$email','$cmfpassword')";

            $result = mysqli\_query($conn, $sql);

    if ($result){

        echo "<script>alert ('Welcome');document.location.href='./index.php'</script>";

        echo "<script>window.open('', '\_self', '');window.close()</script>";

        //mysqli\_free\_result($result);

    }

    //mysqli\_free\_result($result);

?>

**FORGOT PASSWORD (HTML):**

<html>

    <head>

        <title>Forgot Password Form</title>

        <link rel="stylesheet" href="forgot pass.css">

        <body class="main-bg " background="for pass 01.png" style=" background-size: 100%; background-repeat: no-repeat; background-position: center center;background-size: cover;">

            <div class="main-wrapper">

    </head>

    <body>

      <h1>Forgot your password?</h1>

      <hr></hr>

      <h3>Enter your email address to reset your password</h3>

      <form action="index.html" method="post">

        <label for="mail">Email</label></br>

        <input type="email" id="name" name="name" placeholder="Enter your email address" required onblur="validateName(name)">

      <button type="submit">Submit</button>

  <span id="nameError" style="display: none;" >There was an error with your email</span>

      </form>

      <script src="for pass.js"></script>

    </body>

</html>

**FORGOT PASSWORD (CSS):**

$text-color: #30373B;

$success-color: #7AB55C;

$error-color: #C23628;

$border-color: #A7A7A7;

$body-font-size: 13px;

$title-font-size: 32px;

$subtitle-font-size: 20px;

$form-font-size: 16px;

\*,

\*:before,

\*:after {

  -moz-box-sizing: border-box;

  -webkit-box-sizing: border-box;

  box-sizing: border-box;

}

hr {

  background: #c94b4b;

  height: 1px;

  border: 0;

  border-top: 2px solid #ccc;

  padding: 0;

  text-align: right;

  width: 7%;

  float: inline-end;

}

span{

  color: red;

}

label {

  padding-top: 15px;

  font-weight: bold;

}

body {

  font-size: 8px;

  font-family: 'Nunito', sans-serif;

  color: #000000;

}

form {

  font-size: 16px;

  max-width: 200px;

  margin: 10px auto;

  padding: 10px 20px;

  background: #f4f7f8;

  border-radius: 0px;

}

h1 {

  padding-top: 10em;

  font-size: 25px;

  margin: 0 0 25px 0;

  text-align: center;

}

h3 {

  padding-top: 1em;

  font-size: 15px;

  margin: 0 0 30px 0;

  text-align: center;

}

input[type="text"],

input[type="password"],

input[type="date"],

input[type="datetime"],

input[type="email"],

input[type="number"],

input[type="search"],

input[type="tel"],

input[type="time"],

input[type="url"],

textarea,

select {

  background: rgba(255,255,255,0.1);

  border: none;

  font-size: 16px;

  height: auto;

  margin: 0;

  outline: 0;

  padding: 15px;

  width: 66%;

  background-color: #e8eeef;

  color: #8a97a0;

  box-shadow: 0 1px 0 rgba(0, 0, 0, 0.03) inset;

  margin-bottom: 30px;

}

button {

  padding: 12px 39px 13px 39px;

  color: #FFF;

  background-color: #c94b4b;

  font-size: 18px;

  text-align: center;

  font-style: normal;

  border: 1px solid #c94b4b;

  //border-width: 1px 1px 3px;

  margin-bottom: 10px;

  overflow: hidden;

}

label {

  display: block;

  margin-bottom: 8px;

}

@media screen and (min-width: 480px) {

  form {

    max-width: 480px;

  }

}

**FORGOT PASSWORD (JS):**

function validateName(x)

{

  var re = /[A-Za-z@0-9.]/

      if(re.test(document.getElementById(x).value)){

      return true;

    }else{

      // document.getElementById(x ).style.background ='#e35152';

      document.getElementById(x + 'Error').style.display = "block";

      return false;

    }

}

# CONCLUSION

**The Fuel Management System** is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.