Table of Contents

[FUEL DELIVERY ON DEMAND 3](#_Toc124261468)

[ABSTRACT 3](#_Toc124261469)

[CHAPTER 1 – INTRODUCTION 4](#_Toc124261470)

[1.1 System Specifications 5](#_Toc124261471)

[CHAPTER 2 – LITERATURE REVIEW 6](#_Toc124261472)

[CHAPTER 3 OVERALL DESCRIPTION OF THE PROPOSED SYSTEM 9](#_Toc124261473)

[3.1 Module Description 9](#_Toc124261474)

[3.2 System Features 10](#_Toc124261475)

[CHAPTER 4 – DESIGN 11](#_Toc124261476)

[4.1UML Diagrams: 11](#_Toc124261477)

[4.1.1Usecase Diagrams: 11](#_Toc124261478)

[4.1.2 Sequence Diagram: 13](#_Toc124261479)

[4.1.3 Collaboration Diagram: 15](#_Toc124261480)

[4.1.4 Architecture Design 16](#_Toc124261481)

[4.1.5. Data Flow Diagram 16](#_Toc124261482)

[4.1.6. Class Diagram 18](#_Toc124261483)

[4.1.7Table Design 18](#_Toc124261484)

[4.1.8 Work Flow Diagram 20](#_Toc124261485)

[4.1.9 ER Design 20](#_Toc124261486)

[4.1.10 Activity Diagram 21](#_Toc124261487)

[CHAPTER 5 - OUTPUT SCREENSHOTS 21](#_Toc124261488)

[CHAPTER 6 – IMPLEMENTATION DETAILS 23](#_Toc124261489)

[6.1 Introduction to Html Framework 23](#_Toc124261490)

[6.2 Cascading Style Sheets (CSS) 26](#_Toc124261491)

[6.3 MYSQL Server 31](#_Toc124261492)

[6.4PHP 32](#_Toc124261493)

[6.5 ANGULAR JAVA SCRIPT 34](#_Toc124261494)

[CHAPTER 8-TECHNICAL FEASIBILITY 37](#_Toc124261495)

[CHAPTER 9-SYSTEM TESTING 40](#_Toc124261496)

[CHAPTER 10– CONCLUSIONS & FUTURE WORK 44](#_Toc124261497)

[CHAPTER 11- REFERENCES 45](#_Toc124261498)

[CHAPTER 12- CODING 46](#_Toc124261499)

# FUEL DELIVERY ON DEMAND

# ABSTRACT

The **Fuel Delivery on Demand** application to develop delivery ondemand fuel depends on the user order and request. Due to growth of automobiles in market, fuel consumption became more. In existing systm, 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.

# CHAPTER 1 – INTRODUCTION

 The fuel delivery services should comply with enough safety and security measures and should have extensive control in processing the services. For this, you need to set up a system that helps you with data reporting and analysis and do the assessment of different numbers and details that are important in business. In an on-demand fuel delivery service, you may need to measure and control things like temperature and pressure from a security perspective. Further, it also helps in managing inventory and supply and aids in planning marketing activities.

On-demand fuel delivery apps help users to get fuel for their vehicles right at their doorsteps or anywhere they want. Users can simply register on the app, share their location, and request fuel. Once done, the fuel truck will be there at the user’s location within a few minutes. Therefore, with on-demand fuel delivery apps, there will not anymore be frequent stops at the gas station, and rather more of productive time**.** In this application designed with detailing to reflect high quality, shorter delivery cycle, value for money, and expert level understanding of the user’s need.

## System Specifications

**Hardware Requirements:-**

Processor : Intel(R) 2.10GHz

Installed memory (RAM) : 4 GB

Hard Disk : 160 GB

Operating System : Windows (7)

**Software Requirements: -**

Front End: HTML5, CSS3, Bootstrap

Back End: PHP, MYSQL

Control End: Angular Java Script

**Android Tools:**

Android Emulator

xampp-win64 – 8.1

Android Studio

# CHAPTER 2 – LITERATURE REVIEW

**Centralized automation of petrol bunk management and safety using RFID and GSM technology**

The petro-card is swiped on the petro-card reader provided. On swiping, the information is prompted at each step, like password and quantity of petrol in liters. The authentication of the password and checking for sufficiency of balance available for payment is done. If both are fine, the fuel filling gets initiated automatically through the fuel pipe connected to the pump. After the exact quantity of fuel that is entered by the customer is reached, the pumping of fuel gets stopped. In case of wrong password, error message is thrown. Petro-card is recharged with money just like SIM card. In case of low balance amount in the card, online recharge facility is provided to customer, through his smart phone, by just sending SMS to the recharge center GSM module at the bunk. In addition to automated fuel filling, level and smoke detection are also done in this proposed project. A level senor is used to detect the fuel level in the petrol tank and “Low Fuel Level” message is sent to the bunk owner's cell phone. A smoke senor along with buzzer alarm is installed at the bunk, to detect accidental fire. A “smoke detected” alert message is sent to bunk owner's smart phone.

**Computerized filling station management system**

By following this mechanism, we can avoid all the errors that a human does and also avoid the cheating activities that a culprit laborer performs in his work. Our project idea is to make the entire process performed by human laborer in a Filling Station into automated digitalized mechanism to avoid small errors and cheats that can be performed by the laborer to their owners. Our system consists of RFID READER to be placed in the Filling Station and all the vehicles must be provided with RFID TAGS. When the RFID READER senses the RFID TAG that holds the customer id, it sends the id to the centralized server, verifies the ID and opens the gate, the database in the centralized server consist of all banking details of the customer that can be fetched automatically after the confirmation of the user for the cost of the fuel.

At present using RFID technology in automating the filling station including the transaction process have created the idea that it is not secure, so we use advanced cryptographic techniques like Secure Socket Layer (SSL) cryptography. AES - Advanced Encryption Standard Symmetric Cryptography to make sure that the process is secured and efficient. All the customer's ID details and total amount of fuel dispensed from the filling station will also be stored in a separate database for the filling station owner's view and also sent to user's registered mobile number using GSM technology.

**Multi-Atomized Fuel Pump with User Security** In current days fuel stations are operated manually. These fuel pumps are time consuming and require more manpower. To place fuel stations in distant area it very costly to provide excellent facility to the consumers all these problem are sorted out by the use of unmanned petrol pump which requires less time to operate and it is effective and can be installed anywhere the customer self-going to avail the services the payment is done by electronic clearing system. The simple and proper use of microcontroller and GSM technology provides a total security and atomization in distribution of fuel. It has easy operated mobile phone system and graphics user interface (GUI). It is interface with high speed fuel dispenser which is convenient for consumer to operate. In our system the password will be provided to the user via his mobile phone by the petrol pump GSM customer has to enter this password on the LCD provided by the fuel station which will help the petrol company to create authentication for user also the distribution of the fuel is not possible until it gets verified by the database. In short we provide secure system for fuel distribution. The advancement of this project can help industry financially.

**2.1Existing Solution:**

The existing system is the manual system. All the records are maintained in the project. Need to be converted into automated system. In People should push the vehicles or get help to reach nearest gas station. In the above method time and manual work is done by owners of the vehicle. For some aged people or medically ill people it will get even hard. To get fuel to fill generators people need to go to a petrol station.

• Lack of privacy

• Risk in the management of the data.

• Less Security

• Low co-ordination between

• Less User-friendly

• Accuracy not guaranteed

• Not in reach of distant users.

**2.2 Proposed Solution:**

The Project has several features and easy to manage.

1. The System is user friendly.

2. Cost effective.

3. Back up support.

4. Secured Data.

We bring a new solution for refueling automobiles and power backup supply. To develop an application to deliver fuel on demand. To make sure that quality and quantity is good. In this application provides a door to door supply.

In this system the end user will have advantage to select type of fuel required, order and get the fuel in the place they are standing, which makes the process easier for the user.

# CHAPTER 3 OVERALL DESCRIPTION OF THE PROPOSED SYSTEM

## 3.1 Module Description

The system after careful analysis has been identified to be presented with the following modules **User,** **Fuel Station, Admin.**

**User**

* **Register** – User has to register their basic details to get access with this application service.
* **Login** – Once they have registered they need to login to avail the service at the needy time.
* **Search Fuel Station** – to search verified fuel station details based on user need.
* **Place order** – the application will provide you the lists of information user can order the fuel.
* **Make Payment** – An application is that of Payment Integration. The user to pay via different platforms.
* **Get Delivery –**This feature also leaves a lasting impact when it comes to the delivery schedule. Users can request to for a specific time and this allows users to do their other tasks while their fuel tank is being filled.

**Fuel Station**

* **Register** – User has to register their basic details to get access with this application service.
* **Login** – Once they have registered they need to login to avail the service at the needy time.
* **Create Fuel Station** – To create the fuel station enters all the details of a station category. All these information will be stored in the database.
* **Update/Delete Fuel Station –** To update the verified fuel station enters all the details of a they like to station category. All these information will be stored in the database. It can delete the fuel station.
* **Receive order –** To all the area of including the user details they like to station category. All these information will be stored in the database.
* **Approve Status –** To send administrator for verification.

**Admin**

* **Login** – Enter username & password into the login page.
* **Approve/Reject Fuel Station –**Admin only accept whether it is verified fuel station or not fuel station.

## 3.2 System Features

In this application an on-demand fuel delivery application for its client in Australia. This application is developed for Android. This on-demand fuel delivery application will serve as a platform for all the users and corporates to get fuel delivered from the truck drivers that will have trucks fully loaded with fuel.

**System Modules**

**User modules:**

* Register
* Login
* Search Fuel Station
* Place order
* Make Payment
* Get Delivery

**Fuel Station modules:**

* Register
* Login
* Create Fuel Station
* Update/Delete Fuel Station
* Receive order
* Approve Status

**Admin modules:**

* Login
* Approve/Reject Fuel Station

# CHAPTER 4 – DESIGN

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

## 4.1UML Diagrams:

UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

There are various kinds of methods in software design:

* Use case Diagram
* Sequence Diagram
* Collaboration Diagram

## 4.1.1Usecase Diagrams:

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what’s called an actor. Use case diagram can be useful for getting an overall view of the system and clarifying that can do and more importantly what they can’t do.







Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

* The purpose is to show the interactions between the use case and actor.
* To represent the system requirements from user’s perspective.
* An actor could be the end-user of the system or an external system.

## 4.1.2 Sequence Diagram:

Sequence diagram and collaboration diagram are called INTERACTION DIAGRAMS. An interaction diagram shows an interaction, consisting of set of objects and their relationship including the messages that may be dispatched among them.

A sequence diagram is an introduction that empathizes the time ordering of messages. Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages ordered in increasing time along the Y-axis.



## 4.1.3 Collaboration Diagram:

A **collaboration diagram** is a type of visual presentation that shows how various software objects interact with each other within an overall IT architecture and how users can benefit from this **collaboration**. A **collaboration diagram** often comes in the form of a visual chart that resembles a flow chart.



## 4.1.4 Architecture Design

Create Fuel Station

Place order

Customer search fuel station

Update/Delete Fuel Station

User and Fuel Station (Register/ Login)

Make payment

Data base

Approve/Reject Fuel Station

## 4.1.5. Data Flow Diagram

**User**

User

Database

Place Order

Search Fuel Station

Make Payment & Get delivery

**Fuel Station**

Admin

Database

Approve Status

Update/Delete Fuel Station

Create Fuel Station

Receive order

**Admin**

Admin

Database

Approve/Reject Fuel Station

## 4.1.6. Class Diagram



## 4.1.7Table Design

**User Register & Login**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User ID | Name | Email Id | Password | Mobile | Address | City | Question 1 | Question 2 |
| Int | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Primary key |  |  |  |  |  |  |  |  |

**Bunk Register & Login**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User ID | Name | Email Id | | Password | | Mobile | | Address | | City | Question 1 | | Question 2 | |  | |
| Int | Varchar | Varchar | Varchar | | Varchar | | Varchar | | Varchar | | | Varchar | | Varchar | |
| 100 | 100 | 100 | 100 | | 100 | | 100 | | 100 | | | 100 | | 100 | |
| Primary key |  |  |  | |  | |  | |  | | |  | |  | |

**Post Bunk Details**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User ID | | Business Name | | Location | | Landmark | | Mobile | | Mobile 2 | | Address | | City | Geo Location |
| Int | Varchar | | Varchar | | Varchar | | Varchar | | Varchar | | Varchar | | Varchar | |
| 100 | 100 | | 100 | | 100 | | 100 | | 100 | | 100 | | 100 | |
| Primary key |  | |  | |  | |  | |  | |  | |  | |

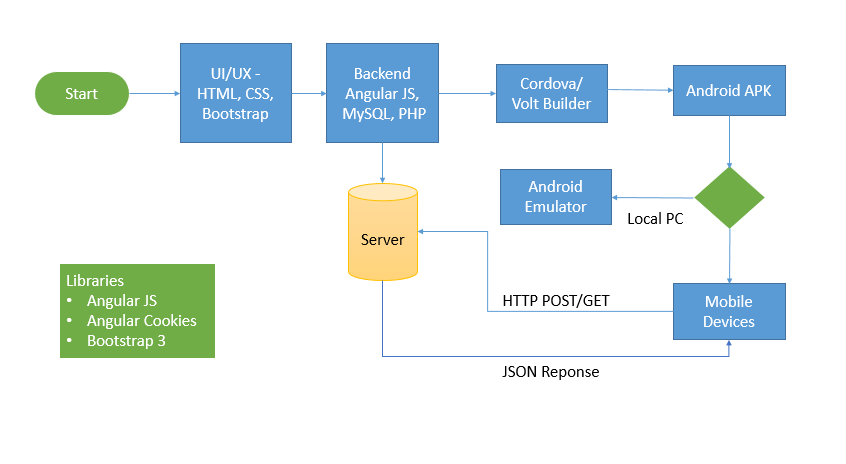
**Booking Status**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User ID | | Business Name | | Card Details | | Exp Date | | CVV | | Mobile | | Qty | Price | Email | |
| Int | Varchar | | Varchar | | Varchar | | Varchar | | Varchar | | Varchar | | Varchar | |
| 100 | 100 | | 100 | | 100 | | 100 | | 100 | | 100 | | 100 | |
| Primary key |  | |  | |  | |  | |  | |  | |  | |

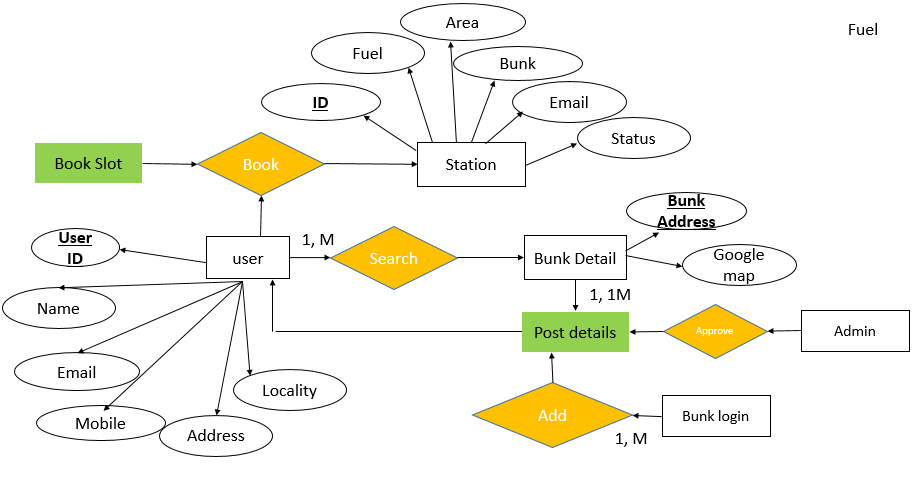
Admin Login

|  |  |  |
| --- | --- | --- |
| User ID | Email | Password |
| Int | Varchar |  |
| 100 | 100 |  |
| Primary key |  |  |

## 4.1.8 Work Flow Diagram



## 4.1.9 ER Design



## 4.1.10 Activity Diagram



# CHAPTER 5 - OUTPUT SCREENSHOTS

# 

# CHAPTER 6 – IMPLEMENTATION DETAILS

## 6.1 Introduction to Html Framework

Hyper Text Markup Language, commonly referred to as HTML, is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) used to create [web pages](https://en.wikipedia.org/wiki/Web_page). Along with [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets), and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), HTML is a cornerstone technology used to create web pages, as well as to create user interfaces for mobile and [web applications](https://en.wikipedia.org/wiki/Web_applications). [Web browsers](https://en.wikipedia.org/wiki/Web_browser) can read HTML files and render them into visible or audible web pages. HTML describes the structure of a [website](https://en.wikipedia.org/wiki/Website) [semantically](https://en.wikipedia.org/wiki/Semantic) along with cues for presentation, making it a markup language, rather than a [programming language](https://en.wikipedia.org/wiki/Programming_language).

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) form the building blocks of HTML pages. HTML allows [images](https://en.wikipedia.org/wiki/Img_(HTML_element)) and other objects to be embedded and it can be used to create [interactive forms](https://en.wikipedia.org/wiki/Fieldset). It provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural semantics for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes and other items. HTML elements are delineated by tags, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets). Tags such as <img /> and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed [scripts](https://en.wikipedia.org/wiki/Scripting_language) written in languages such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript) which affect the behavior of HTML web pages. HTML markup can also refer the browser to [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) to define the look and layout of text and other material.

HyperText Markup Language (HTML) is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for creating [web pages](https://en.wikipedia.org/wiki/Web_page) and [web applications](https://en.wikipedia.org/wiki/Web_application). With [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [JavaScript](https://en.wikipedia.org/wiki/JavaScript) it forms a triad of cornerstone technologies for the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).[[1]](https://en.wikipedia.org/wiki/HTML#cite_note-1) [Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [webserver](https://en.wikipedia.org/wiki/Webserver) or from local storage and render them into multimedia web pages. HTML describes the structure of a web page [semantically](https://en.wikipedia.org/wiki/Semantic) and originally included cues for the appearance of the document.

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) are the building blocks of HTML pages. With HTML constructs, [images](https://en.wikipedia.org/wiki/Img_%28HTML_element%29) and other objects, such as [interactive forms,](https://en.wikipedia.org/wiki/Fieldset) may be embedded into the rendered page. It provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](https://en.wikipedia.org/wiki/Semantics) for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes and other items. HTML elements are delineated by tags, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets). Tags such as <img /> and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a [scripting language](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript) which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[[2]](https://en.wikipedia.org/wiki/HTML#cite_note-deprecated-2)

In 1980, physicist [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee), a contractor at [CERN](https://en.wikipedia.org/wiki/CERN), proposed and prototyped [ENQUIRE](https://en.wikipedia.org/wiki/ENQUIRE), a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an [Internet](https://en.wikipedia.org/wiki/Internet)-based [hypertext](https://en.wikipedia.org/wiki/Hypertext) system.[[3]](https://en.wikipedia.org/wiki/HTML#cite_note-3) Berners-Lee specified HTML and wrote the browser and server software in late 1990. That year, Berners-Lee and CERN data systems engineer [Robert Cailliau](https://en.wikipedia.org/wiki/Robert_Cailliau) collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes[[4]](https://en.wikipedia.org/wiki/HTML#cite_note-4) from 1990 he listed[[5]](https://en.wikipedia.org/wiki/HTML#cite_note-5) "some of the many areas in which hypertext is used" and put an encyclopedia first.

The first publicly available description of HTML was a document called "HTML Tags", first mentioned on the Internet by Tim Berners-Lee in late 1991.[[6]](https://en.wikipedia.org/wiki/HTML#cite_note-tagshtml-6)[[7]](https://en.wikipedia.org/wiki/HTML#cite_note-7) It describes 18 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by [SGMLguid](https://en.wikipedia.org/wiki/SGMLguid), an in-house [Standard Generalized Markup Language](https://en.wikipedia.org/wiki/Standard_Generalized_Markup_Language) (SGML)-based documentation format at CERN. Eleven of these elements still exist in HTML 4.[[8]](https://en.wikipedia.org/wiki/HTML#cite_note-8)

HTML is a [markup language](https://en.wikipedia.org/wiki/Markup_language) that [web browsers](https://en.wikipedia.org/wiki/Web_browser) use to interpret and [compose](https://en.wikipedia.org/wiki/Typesetting) text, images, and other material into visual or audible web pages. Default characteristics for every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets). Many of the text elements are found in the 1988 ISO technical report TR 9537 Techniques for using SGML, which in turn covers the features of early text formatting languages such as that used by the [RUNOFF command](https://en.wikipedia.org/wiki/TYPSET_and_RUNOFF) developed in the early 1960s for the [CTSS](https://en.wikipedia.org/wiki/Compatible_Time-Sharing_System) (Compatible Time-Sharing System) operating system: these formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with also the separation of structure and markup; HTML has been progressively moved in this direction with CSS.

Berners-Lee considered HTML to be an application of SGML. It was formally defined as such by the [Internet Engineering Task Force](https://en.wikipedia.org/wiki/Internet_Engineering_Task_Force) (IETF) with the mid-1993 publication of the first proposal for an HTML specification, the "Hypertext Markup Language (HTML)" Internet Draft by Berners-Lee and [Dan Connolly](https://en.wikipedia.org/wiki/Dan_Connolly_%28computer_scientist%29), which included an SGML [Document Type Definition](https://en.wikipedia.org/wiki/Document_Type_Definition) to define the grammar.[[9]](https://en.wikipedia.org/wiki/HTML#cite_note-9)[[10]](https://en.wikipedia.org/wiki/HTML#cite_note-10) The draft expired after six months, but was notable for its acknowledgment of the [NCSA Mosaic](https://en.wikipedia.org/wiki/Mosaic_%28web_browser%29) browser's custom tag for embedding in-line images, reflecting the IETF's philosophy of basing standards on successful prototypes.[[11]](https://en.wikipedia.org/wiki/HTML#cite_note-raymond-11) Similarly, [Dave Raggett](https://en.wikipedia.org/wiki/Dave_Raggett)'s competing Internet-Draft, "HTML+ (Hypertext Markup Format)", from late 1993, suggested standardizing already-implemented features like tables and fill-out forms.[[12]](https://en.wikipedia.org/wiki/HTML#cite_note-html.2B-12)

After the HTML and HTML+ drafts expired in early 1994, the IETF created an HTML Working Group, which in 1995 completed "HTML 2.0", the first HTML specification intended to be treated as a standard against which future implementations should be based.[[13]](https://en.wikipedia.org/wiki/HTML#cite_note-13)

Further development under the auspices of the IETF was stalled by competing interests. Since 1996, the HTML specifications have been maintained, with input from commercial software vendors, by the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C).[[14]](https://en.wikipedia.org/wiki/HTML#cite_note-raggett-14) However, in 2000, HTML also became an international standard ([ISO](https://en.wikipedia.org/wiki/International_Organization_for_Standardization)/[IEC](https://en.wikipedia.org/wiki/International_Electrotechnical_Commission) 15445:2000). HTML 4.01 was published in late 1999, with further errata published through 2001. In 2004, development began on HTML5 in the [Web Hypertext Application Technology Working Group](https://en.wikipedia.org/wiki/Web_Hypertext_Application_Technology_Working_Group) (WHATWG), which became a joint deliverable with the W3C in 2008, and completed and standardized on 28 October 2014.[[15]](https://en.wikipedia.org/wiki/HTML#cite_note-15)

## 6.2 Cascading Style Sheets (CSS)

CSS is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language). Although most often used to set the visual style of [web pages](https://en.wikipedia.org/wiki/Web_page) and user interfaces written in [HTML](https://en.wikipedia.org/wiki/HTML) and [XHTML](https://en.wikipedia.org/wiki/XHTML), the language can be applied to any [XML](https://en.wikipedia.org/wiki/XML) document, including [plain XML](https://en.wikipedia.org/wiki/Plain_Old_XML), [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics) and[XUL](https://en.wikipedia.org/wiki/XUL), and is applicable to rendering in [speech](https://en.wikipedia.org/wiki/Speech_synthesis), or on other media. Along with HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for [web applications](https://en.wikipedia.org/wiki/Web_applications), and user interfaces for many mobile applications.

CSS is designed primarily to enable [the separation of document content from document presentation](https://en.wikipedia.org/wiki/Separation_of_presentation_and_content), including aspects such as the [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface). This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as [semantically insignificant tables](https://en.wikipedia.org/wiki/Tableless_web_design) that were widely used to format pages before consistent CSS rendering was available in all major browsers. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. For each matching [HTML element](https://en.wikipedia.org/wiki/HTML_element), it provides a list of formatting instructions. For example, a CSS rule might specify that "all heading 1 elements should be [bold](https://en.wikipedia.org/wiki/Bold)", leaving pure semantic HTML markup that asserts "this text is a level 1 heading" without formatting code such as a<bold> tag indicating how such text should be displayed.

This separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or[screen reader](https://en.wikipedia.org/wiki/Screen_reader)) and on [Braille-based](https://en.wikipedia.org/wiki/Braille_display), tactile devices. It can also be used to display the web page differently depending on the screen size or device on which it is being viewed. Although the author of a web page typically links to a CSS file within the markup file, readers can specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author has specified. If the author or the reader did not link the document to a style sheet, the default style of the browser will be applied. Another advantage of CSS is that aesthetic changes to the [graphic design](https://en.wikipedia.org/wiki/Graphic_design) of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in one file, rather than by a laborious (and thus expensive) process of crawling over every document line by line, changing markup.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable.

Cascading Style Sheets (CSS) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language).[[1]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-1) Although most often used to set the visual style of [web pages](https://en.wikipedia.org/wiki/Web_page) and user interfaces written in [HTML](https://en.wikipedia.org/wiki/HTML) and [XHTML](https://en.wikipedia.org/wiki/XHTML), the language can be applied to any [XML](https://en.wikipedia.org/wiki/XML) document, including [plain XML](https://en.wikipedia.org/wiki/Plain_Old_XML), [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics) and [XUL](https://en.wikipedia.org/wiki/XUL), and is applicable to rendering in [speech](https://en.wikipedia.org/wiki/Speech_synthesis), or on other media. Along with HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for [web applications](https://en.wikipedia.org/wiki/Web_applications), and user interfaces for many mobile applications.[[2]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-2)

CSS is designed primarily to enable [the separation of document content from document presentation](https://en.wikipedia.org/wiki/Separation_of_presentation_and_content), including aspects such as the [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface).[[3]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-3) This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or [screen reader](https://en.wikipedia.org/wiki/Screen_reader)), and on [Braille-based](https://en.wikipedia.org/wiki/Braille_display) tactile devices. It can also display the web page differently depending on the screen size or viewing device. Readers can also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author specified.

Changes to the [graphic design](https://en.wikipedia.org/wiki/Graphic_design) of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing markup in the documents.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C). Internet media type ([MIME type](https://en.wikipedia.org/wiki/MIME_media_type)) text/css is registered for use with CSS by [RFC 2318](https://tools.ietf.org/html/rfc2318) (March 1998). The W3C operates a free [CSS validation service](https://en.wikipedia.org/wiki/W3C_Markup_Validation_Service#CSS_validation) for CSS documents.

In CSS, selectors declare which part of the markup a style applies to by matching tags and attributes in the markup itself.

Selectors may apply to:

all [elements](https://en.wikipedia.org/wiki/HTML_element) of a specific type, e.g. the second-level headers [h2](https://en.wikipedia.org/wiki/HTML_element#Basic_text)

elements specified by [attribute](https://en.wikipedia.org/wiki/HTML_attribute), in particular:

id: an identifier unique within the document

class: an identifier that can annotate multiple elements in a document

elements depending on how they are placed relative to others in the [document tree](https://en.wikipedia.org/wiki/Document_Object_Model).

Classes and IDs are case-sensitive, start with letters, and can include alphanumeric characters and underscores. A class may apply to any number of instances of any elements. An ID may only be applied to a single element.

Pseudo-classes are used in CSS selectors to permit formatting based on information that is not contained in the document tree. One example of a widely used pseudo-class is :hover, which identifies content only when the user "points to" the visible element, usually by holding the mouse cursor over it. It is appended to a selector as in a:hover or #elementid:hover. A pseudo-class classifies document elements, such as :link or :visited, whereas a pseudo-element makes a selection that may consist of partial elements, such as ::first-line or ::first-letter.[[5]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-5)

Selectors may be combined in many ways to achieve great specificity and flexibility.[[6]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-6) Multiple selectors may be joined in a spaced list to specify elements by location, element type, id, class, or any combination thereof. The order of the selectors is important. For example, div .myClass {color: red;} applies to all elements of class myClass that are inside div elements, whereas .myClass div {color: red;} applies to all div elements that are in elements of class myClass.

CSS information can be provided from various sources. These sources can be the web browser, the user and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium.

The style sheet with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style. This process is called cascading.

One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

CSS was first proposed by [Håkon Wium Lie](https://en.wikipedia.org/wiki/H%C3%A5kon_Wium_Lie) on October 10, 1994.[[16]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chss-proposal-16) At the time, Lie was working with [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee) at [CERN](https://en.wikipedia.org/wiki/CERN).[[17]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chapter20-17) Several other style sheet languages for the web were proposed around the same time, and discussions on public mailing lists and inside [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) resulted in the first W3C CSS Recommendation (CSS1)[[18]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-w3c-css1-18) being released in 1996. In particular, [Bert Bos](https://en.wikipedia.org/wiki/Bert_Bos)' proposal was influential; he became co-author of CSS1 and is regarded as co-creator of CSS.[[19]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-WWW3-19)

Style sheets have existed in one form or another since the beginnings of Standard Generalized Markup Language ([SGML](https://en.wikipedia.org/wiki/SGML)) in the 1980s, and CSS was developed to provide style sheets for the web.[[20]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-css-phd-20) One requirement for a web style sheet language was for style sheets to come from different sources on the web. Therefore, existing style sheet languages like [DSSSL](https://en.wikipedia.org/wiki/Document_Style_Semantics_and_Specification_Language) and [FOSI](https://en.wikipedia.org/wiki/Formatting_Output_Specification_Instance) were not suitable. CSS, on the other hand, let a document's style be influenced by multiple style sheets by way of "cascading" styles.[[20]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-css-phd-20)

As [HTML](https://en.wikipedia.org/wiki/HTML) grew, it came to encompass a wider variety of stylistic capabilities to meet the demands of [web developers](https://en.wikipedia.org/wiki/Web_development). This evolution gave the designer more control over site appearance, at the cost of more complex HTML. Variations in [web browser](https://en.wikipedia.org/wiki/Web_browser) implementations, such as [ViolaWWW](https://en.wikipedia.org/wiki/ViolaWWW) and [WorldWideWeb](https://en.wikipedia.org/wiki/WorldWideWeb),[[21]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-IEEE-21) made consistent site appearance difficult, and users had less control over how web content was displayed. The browser/editor developed by Tim Berners-Lee had style sheets that were hard-coded into the program. The style sheets could therefore not be linked to documents on the web.[[22]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-cssdftw-22) [Robert Cailliau](https://en.wikipedia.org/wiki/Robert_Cailliau), also of CERN, wanted to separate the structure from the presentation so that different style sheets could describe different presentation for printing, screen-based presentations, and editors.[[21]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-IEEE-21)

Improving web presentation capabilities was a topic of interest to many in the web community and nine different style sheet languages were proposed on the www-style mailing list.[[20]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-css-phd-20) Of these nine proposals, two were especially influential on what became CSS: Cascading HTML Style Sheets[[16]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chss-proposal-16) and Stream-based Style Sheet Proposal (SSP).[[19]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-WWW3-19)[[23]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-ssp-23) Two browsers served as testbeds for the initial proposals; Lie worked with [Yves Lafon](https://en.wikipedia.org/w/index.php?title=Yves_Lafon&action=edit&redlink=1) to implement CSS in [Dave Raggett](https://en.wikipedia.org/wiki/Dave_Raggett)'s [Arena](https://en.wikipedia.org/wiki/Arena_%28web_browser%29) browser.[[24]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-24)[[25]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-25)[[26]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-26) [Bert Bos](https://en.wikipedia.org/wiki/Bert_Bos) implemented his own SSP proposal in the [Argo](https://en.wikipedia.org/wiki/Argo_%28web_browser%29) browser.[[19]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-WWW3-19) Thereafter, Lie and Bos worked together to develop the CSS standard (the 'H' was removed from the name because these style sheets could also be applied to other markup languages besides HTML).[[17]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chapter20-17)

Lie's proposal was presented at the "[Mosaic and the Web](https://en.wikipedia.org/w/index.php?title=Mosaic_and_the_Web&action=edit&redlink=1)" conference (later called WWW2) in [Chicago, Illinois](https://en.wikipedia.org/wiki/Chicago,_Illinois) in 1994, and again with Bert Bos in 1995.[[17]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chapter20-17) Around this time the W3C was already being established, and took an interest in the development of CSS. It organized a workshop toward that end chaired by [Steven Pemberton](https://en.wikipedia.org/wiki/Steven_Pemberton). This resulted in W3C adding work on CSS to the deliverables of the HTML editorial review board (ERB). Lie and Bos were the primary technical staff on this aspect of the project, with additional members, including [Thomas Reardon](https://en.wikipedia.org/wiki/Thomas_Reardon) of [Microsoft](https://en.wikipedia.org/wiki/Microsoft), participating as well. In August 1996 [Netscape Communication Corporation](https://en.wikipedia.org/wiki/Netscape) presented an alternative style sheet language called [JavaScript Style Sheets](https://en.wikipedia.org/wiki/JavaScript_Style_Sheets) (JSSS).[[17]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chapter20-17) The spec was never finished and is deprecated.[[27]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-27) By the end of 1996, CSS was ready to become official, and the CSS level 1 Recommendation was published in December.

Development of HTML, CSS, and the [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) had all been taking place in one group, the HTML Editorial Review Board (ERB). Early in 1997, the ERB was split into three [working groups](https://en.wikipedia.org/wiki/Working_group): [HTML Working group](https://en.wikipedia.org/wiki/HTML_Working_group), chaired by [Dan Connolly](https://en.wikipedia.org/wiki/Dan_Connolly_%28computer_scientist%29) of W3C; DOM Working group, chaired by Lauren Wood of [SoftQuad](https://en.wikipedia.org/wiki/SoftQuad); and [CSS Working group](https://en.wikipedia.org/wiki/CSS_Working_group), chaired by [Chris Lilley](https://en.wikipedia.org/wiki/Chris_Lilley_%28W3C%29) of W3C.

The CSS Working Group began tackling issues that had not been addressed with CSS level 1, resulting in the creation of CSS level 2 on November 4, 1997. It was published as a W3C Recommendation on May 12, 1998. CSS level 3, which was started in 1998, is still under development as of 2014.

In 2005 the CSS Working Groups decided to enforce the requirements for standards more strictly. This meant that already published standards like CSS 2.1, CSS 3 Selectors and CSS 3 Text were pulled back from Candidate Recommendation to Working Draft level.

## 6.3 MYSQL Server

MySQL  is an [open-source](https://en.wikipedia.org/wiki/Open-source) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS);[[6]](https://en.wikipedia.org/wiki/MySQL#cite_note-6) in July 2013, it was the world's second most widely used RDBMS, and the most widely used open-source [client–server model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model) RDBMS. It is named after co-founder [Michael Widenius](https://en.wikipedia.org/wiki/Michael_Widenius)'s daughter, My. The [SQL](https://en.wikipedia.org/wiki/SQL) acronym stands for [Structured Query Language](https://en.wikipedia.org/wiki/Structured_Query_Language). The MySQL development project has made its [source code](https://en.wikipedia.org/wiki/Source_code) available under the terms of the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), as well as under a variety of [proprietary](https://en.wikipedia.org/wiki/Proprietary_software) agreements. MySQL was owned and sponsored by a single [for-profit](https://en.wikipedia.org/wiki/Business) firm, the [Swedish](https://en.wikipedia.org/wiki/Sweden)company [MySQL AB](https://en.wikipedia.org/wiki/MySQL_AB), now owned by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation). For proprietary use, several paid editions are available, and offer additional functionality.

SQL Server Management Studio (SSMS) is a software application first launched with [Microsoft](https://en.wikipedia.org/wiki/Microsoft) [SQL Server 2005](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) that is used for configuring, managing, and administering all components within [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server). The tool includes both script editors and graphical tools which work with objects and features of the server.[[1]](https://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-1)

A central feature of SSMS is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server.[[2]](https://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-2) It also shipped a separate Express edition that could be freely downloaded, however recent versions of SSMS are fully capable of connecting to and manage any SQL Server Express instance. Microsoft also incorporated backwards compatibility for older versions of SQL Server thus allowing a newer version of SSMS to connect to older versions of SQL Server instances.

Starting from version 11, the application was based on the [Visual Studio 2010](https://en.wikipedia.org/wiki/Visual_Studio_2010) shell, using [WPF](https://en.wikipedia.org/wiki/Windows_Presentation_Foundation) for the user interface.

In June 2015, Microsoft announced their intention to release future versions of SSMS independently of SQL Server database engine releases.[[3]](https://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-3).

## 6.4PHP

PHP is a [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting) language designed for [web development](https://en.wikipedia.org/wiki/Web_development) but also used as a [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Originally created by [RasmusLerdorf](https://en.wikipedia.org/wiki/Rasmus_Lerdorf) in 1994, the PHP [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the[recursive](https://en.wikipedia.org/wiki/Recursive_acronym) [backronym](https://en.wikipedia.org/wiki/Backronym) PHP: Hypertext Preprocessor.

PHP code may be embedded into [HTML](https://en.wikipedia.org/wiki/HTML) code, or it can be used in combination with various [web template systems](https://en.wikipedia.org/wiki/Web_template_system), web content management system and [web frameworks](https://en.wikipedia.org/wiki/Web_framework). PHP code is usually processed by a PHP[interpreter](https://en.wikipedia.org/wiki/Interpreter_(computing)) implemented as a [module](https://en.wikipedia.org/wiki/Plugin_(computing)) in the web server or as a [Common Gateway Interface](https://en.wikipedia.org/wiki/Common_Gateway_Interface) (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface)(CLI) and can be used to implement [standalone](https://en.wikipedia.org/wiki/Computer_software) [graphical applications](https://en.wikipedia.org/wiki/Graphical_user_interface).

The standard PHP interpreter, powered by the [Zend Engine](https://en.wikipedia.org/wiki/Zend_Engine), is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [PHP License](https://en.wikipedia.org/wiki/PHP_License). PHP has been widely ported and can be deployed on most web servers on almost every [operating system](https://en.wikipedia.org/wiki/Operating_system) and[platform](https://en.wikipedia.org/wiki/Computing_platform), free of charge.

The PHP language evolved without a written [formal specification](https://en.wikipedia.org/wiki/Formal_specification) or standard until 2014, leaving the canonical PHP interpreter as a [de facto](https://en.wikipedia.org/wiki/De_facto) standard. Since 2014 work has gone on to create a formal PHP specification.

PHP is a [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting) language designed primarily for [web development](https://en.wikipedia.org/wiki/Web_development) but also used as a [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Originally created by [Rasmus Lerdorf](https://en.wikipedia.org/wiki/Rasmus_Lerdorf) in 1994,[[4]](https://en.wikipedia.org/wiki/PHP#cite_note-History_of_PHP-4) the PHP [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) is now produced by The PHP Development Team.[[5]](https://en.wikipedia.org/wiki/PHP#cite_note-about_PHP-5) PHP originally stood for Personal Home Page,[[4]](https://en.wikipedia.org/wiki/PHP#cite_note-History_of_PHP-4) but it now stands for the [recursive acronym](https://en.wikipedia.org/wiki/Recursive_acronym) PHP: Hypertext Preprocessor.[[6]](https://en.wikipedia.org/wiki/PHP#cite_note-6)

PHP code may be embedded into [HTML](https://en.wikipedia.org/wiki/HTML) or HTML5 code, or it can be used in combination with various [web template systems](https://en.wikipedia.org/wiki/Web_template_system), [web content management systems](https://en.wikipedia.org/wiki/Web_content_management_system) and [web frameworks](https://en.wikipedia.org/wiki/Web_framework). PHP code is usually processed by a PHP [interpreter](https://en.wikipedia.org/wiki/Interpreter_%28computing%29) implemented as a [module](https://en.wikipedia.org/wiki/Plugin_%28computing%29) in the web server or as a [Common Gateway Interface](https://en.wikipedia.org/wiki/Common_Gateway_Interface) (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface) (CLI) and can be used to implement [standalone](https://en.wikipedia.org/wiki/Computer_software) [graphical applications](https://en.wikipedia.org/wiki/Graphical_user_interface).[[7]](https://en.wikipedia.org/wiki/PHP#cite_note-7)

The standard PHP interpreter, powered by the [Zend Engine](https://en.wikipedia.org/wiki/Zend_Engine), is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [PHP License](https://en.wikipedia.org/wiki/PHP_License). PHP has been widely ported and can be deployed on most web servers on almost every [operating system](https://en.wikipedia.org/wiki/Operating_system) and [platform](https://en.wikipedia.org/wiki/Computing_platform), free of charge.[[8]](https://en.wikipedia.org/wiki/PHP#cite_note-foundations-8)

The PHP language evolved without a written [formal specification](https://en.wikipedia.org/wiki/Formal_specification) or standard until 2014, leaving the canonical PHP interpreter as a [de facto](https://en.wikipedia.org/wiki/De_facto) standard. Since 2014 work has gone on to create a formal PHP specification.[[9]](https://en.wikipedia.org/wiki/PHP#cite_note-9)

PHP development began in 1995 when [Rasmus Lerdorf](https://en.wikipedia.org/wiki/Rasmus_Lerdorf) wrote several [Common Gateway Interface](https://en.wikipedia.org/wiki/Common_Gateway_Interface) (CGI) programs in C,[[10]](https://en.wikipedia.org/wiki/PHP#cite_note-php_origins-10)[[11]](https://en.wikipedia.org/wiki/PHP#cite_note-11)[[12]](https://en.wikipedia.org/wiki/PHP#cite_note-12) which he used to maintain his [personal homepage](https://en.wikipedia.org/wiki/Personal_homepage). He extended them to work with [web forms](https://en.wikipedia.org/wiki/Web_form) and to communicate with [databases](https://en.wikipedia.org/wiki/Database), and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.

PHP/FI could help to build simple, dynamic [web applications](https://en.wikipedia.org/wiki/Web_application). To accelerate [bug](https://en.wikipedia.org/wiki/Software_bug) reporting and to improve the code, Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP Tools) version 1.0" on the [Usenet](https://en.wikipedia.org/wiki/Usenet) discussion group comp.infosystems.www.authoring.cgi on June 8, 1995.[[13]](https://en.wikipedia.org/wiki/PHP#cite_note-13)[[14]](https://en.wikipedia.org/wiki/PHP#cite_note-14) This release already had the basic functionality that PHP has as of 2013. This included [Perl-like variables](https://en.wikipedia.org/wiki/Local_variable#Local_variables_in_Perl), form handling, and the ability to embed HTML. The [syntax](https://en.wikipedia.org/wiki/Syntax) resembled that of Perl but was simpler, more limited and less consistent.[[5]](https://en.wikipedia.org/wiki/PHP#cite_note-about_PHP-5)

Lerdorf did not intend the early PHP to become a new programming language, but it grew organically, with Lerdorf noting in retrospect: "I don’t know how to stop it, there was never any intent to write a programming language […] I have absolutely no idea how to write a programming language, I just kept adding the next logical step on the way."[[15]](https://en.wikipedia.org/wiki/PHP#cite_note-itconversations-15) A development team began to form and, after months of work and [beta](https://en.wikipedia.org/wiki/Beta_development_stage) testing, officially released PHP/FI 2 in November 1997.

The fact that PHP lacked an original overall design but instead developed organically has led to inconsistent naming of functions and inconsistent ordering of their parameters.[[16]](https://en.wikipedia.org/wiki/PHP#cite_note-16) In some cases, the function names were chosen to match the lower-level libraries which PHP was "wrapping",[[17]](https://en.wikipedia.org/wiki/PHP#cite_note-17) while in some very early versions of PHP the length of the function names was used internally as a [hash function](https://en.wikipedia.org/wiki/Hash_function), so names were chosen to improve the distribution of hash values.[[18]](https://en.wikipedia.org/wiki/PHP#cite_note-php-internals-70691-18)

## 6.5 ANGULAR JAVA SCRIPT

AngularJS (commonly referred to as "Angular" or "Angular.js") is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) [web application framework](https://en.wikipedia.org/wiki/Web_application_framework) mainly maintained by [Google](https://en.wikipedia.org/wiki/Google) and by a community of individuals and corporations to address many of the challenges encountered in developing [single-page applications](https://en.wikipedia.org/wiki/Single-page_application). It aims to simplify both the development and the [testing](https://en.wikipedia.org/wiki/Software_testing) of such applications by providing a framework for client-side [model–view–controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) and [model–view–viewmodel](https://en.wikipedia.org/wiki/Model_View_ViewModel)(MVVM) architectures, along with components commonly used in [rich Internet applications](https://en.wikipedia.org/wiki/Rich_Internet_Application).

The AngularJS framework works by first reading the [HTML](https://en.wikipedia.org/wiki/HTML) page, which has embedded into it additional custom [tag attributes](https://en.wikipedia.org/wiki/HTML_attribute). Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard [JavaScript](https://en.wikipedia.org/wiki/JavaScript) variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic [JSON](https://en.wikipedia.org/wiki/JSON) resources.

According to [JavaScript](https://en.wikipedia.org/wiki/JavaScript) analytics service [Libscore](https://en.wikipedia.org/wiki/Libscore), AngularJS is used on the websites of [Wolfram Alpha](https://en.wikipedia.org/wiki/Wolfram_Alpha), [NBC](https://en.wikipedia.org/wiki/NBC),[Walgreens](https://en.wikipedia.org/wiki/Walgreens), [Intel](https://en.wikipedia.org/wiki/Intel), [Sprint](https://en.wikipedia.org/wiki/Sprint_Nextel), [ABC News](https://en.wikipedia.org/wiki/ABC_News), and approximately 8,400 other sites out of 1 million tested in July 2015.

AngularJS is the frontend part of the [MEAN stack](https://en.wikipedia.org/wiki/MEAN_(software_bundle)), consisting of [MongoDB](https://en.wikipedia.org/wiki/MongoDB) database, [Express.js](https://en.wikipedia.org/wiki/Express.js) web application server framework, Angular.js itself, and [Node.js](https://en.wikipedia.org/wiki/Node.js) runtime environment.

AngularJS is an open source web application framework. It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google. Its latest version is 1.4.3.

Definition of AngularJS as put by its [official documentation](https://docs.angularjs.org/guide/introduction) is as follows −

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

**Features**

* AngularJS is a powerful JavaScript based development framework to create RICH Internet Application(RIA).
* AngularJS provides developers options to write client side application (using JavaScript) in a clean MVC(Model View Controller) way.
* Application written in AngularJS is cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
* AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.
* Overall, AngularJS is a framework to build large scale and high performance web application while keeping them as easy-to-maintain.

**Core Features**

Following are most important core features of AngularJS −

* **Data-binding** − It is the automatic synchronization of data between model and view components.
* **Scope −** These are objects that refer to the model. They act as a glue between controller and view.
* **Controller −** These are JavaScript functions that are bound to a particular scope.
* **Services −** AngularJS come with several built-in services for example $https: to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
* **Filters** − These select a subset of items from an array and returns a new array.
* **Directives −** Directives are markers on DOM elements (such as elements, attributes, css, and more). These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives (ngBind, ngModel...)
* **Templates** − These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".
* **Routing** − It is concept of switching views.
* **Model View Whatever** − MVC is a design pattern for dividing an application into different parts (called Model, View and Controller), each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
* **Deep Linking** − Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
* **Dependency Injection** − AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.

**CHAPTER 7- SYSTEM STUDY**

**7.1 FEASIBILITY 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.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This 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 of 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.

# CHAPTER 8-TECHNICAL FEASIBILITY

This 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. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**8.1Non Functional Requirements**

Non-functional requirements are the quality requirements that stipulate how well software does what it has to do. These are Quality attributes of any system; these can be seen at the execution of the system and they can also be the part of the system architecture.

**8.2 Accuracy:**

The system will be accurate and reliable based on the design architecture. If there is any problem in the accuracy then the system will provide alternative ways to solve the problem.

**8.3 Usability:**

The proposed system will be simple and easy to use by the users. The users will comfort in order to communicate with the system. The user will be provided with an easy interface of the system.

**8.4 Accessibility:**

The system will be accessible through internet and there should be no any known problem.

* 1. **Performance:**

The system performance will be at its best when performing the functionality of the system.

* 1. **Reliability:**

The proposed system will be reliable in all circumstances and if there is any problem that will be affectively handle in the design.

* 1. **Security:**

The proposed system will be highly secured; every user will be required registration and username/password to use the system. The system will do the proper authorization and authentication of the users based on their types and their requirements. The proposed system will be designed persistently to avoid any misuse of the application.

# CHAPTER 9-SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**9.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page

**9.2 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**9.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# 

# CHAPTER 10– CONCLUSIONS & FUTURE WORK

In this android application forms provided to the users should not be complicated as customers want simple solutions. These forms should include details like Fuel type, quantity, additional add-ons and many more. Once, the key features have been covered, next let's find out the additional features that can be integrated into the app to give it a competitive edge.

To distribution our system looks for the control on product thefts which is the most serious problem for the manufacturing industries and reduction in manpower required. It is also possible to implement the same system for milk processing industries while distributing the milk and its products to the market. In day to day life we can see that water distribution in summer is also one of the problems in front of India. So it is possible to keep control on water distribution in particular area. Also it is possible to keep record of the distributed products in market which is commercially most important for industries.

# CHAPTER 11- REFERENCES

[1]. Kulkarni Amruta M. & Taware Sachin S.―Embedded Security System Using RFID & GSM Module‖ (International Journal of Computer Technology & Electronic Engg.) Volume 2 (Issue 1), Page No. 164-168.

[2]. Behera Susanta K. & Ali Farida A. ―Automobile Fuel Pump Control System Using Embedded System‖ (International Journal Of Computer

Technology & Electronic Engg.) Volume 3 (Issue 2), Page No. 41-47. April 2013.

[3]. Kapse Sagar Sudhakar, Abhale Amol Anil, Kudake chetan Ashok,Shirsath Shravan Bhaskar. ―Automatic Street Light Control System‖ (International Journal of Emerging Technology and Advanced Engineering) Volume 3, Issue 5,

May 2013.

[4]. S. K. Singh, ―Industrial Instrumentation & Control‖ Tata McGraw Hill, .246.

[5]. Dr. A.D.Shaligram, ―Sensor &Transducer‖ C.T.C,135.

# CHAPTER 12- CODING

**Angular\_user.js**

var app = angular.module("myapp", ['ngCookies']);

app.controller("myappCtrl", function($scope, $cookieStore, $cookies, $http)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* User Login \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// sign in button

$scope.user\_login = function()

{

$http.post('user\_login.php',

{'email': $scope.email, 'password':$scope.password})

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

alert("Login Successful");

$cookieStore.put("cook\_user\_email",data.email);

$cookieStore.put("cook\_staff\_dept",data.field\_1);

window.location = "user\_home.html"; // Home Page

return;

}

else if(data.success == 2)

{

alert("Please Fill All Fields");

}

else

{

alert("Login Unsuccessful");

}

});

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Cookies \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$scope.cook\_user\_email = $cookieStore.get("cook\_user\_email");

$scope.cook\_staff\_dept = $cookieStore.get("cook\_staff\_dept");

$scope.worker\_register = function()

{

$http.post('worker\_register.php',{

'name':$scope.name,'email':$scope.email,'password':$scope.password,

'mobile': $scope.mobile,'field\_1': $scope.field\_1,'field\_2': $scope.field\_2,

'field\_3': $scope.field\_3})

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

alert("Registered successfully");

window.location = "worker\_login.html";

return;

}

else if(data.success == 2)

{

alert("Please Fill All Fields");

}

else if(data.success == 0)

{

alert("Error");

}

else

{

alert(" Un Successfull");

}

});

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Worker Login \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// sign in button

$scope.worker\_login = function()

{

$http.post('worker\_login.php',

{'email': $scope.email, 'password':$scope.password})

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

alert("Login Successful");

window.location = "home.html";

$cookieStore.put("cook\_work\_email",data.email);

$cookieStore.put("cook\_type",data.field\_1);

// Home Page

return;

}

else if(data.success == 2)

{

alert("Please Fill All Fields");

}

else

{

alert("Login Unsuccessful");

}

});

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* User Logout \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$scope.user\_logout = function()

{

if(confirm("Are You Sure?"))

{

$cookies.cook\_user\_email = "";

$cookies.cook\_admin\_email = "";

window.location = "index.html";

return;

}

else

{

return false;

}

}

$scope.admin\_logout = function()

{

if(confirm("Are You Sure?"))

{

$cookies.cook\_admin\_email = "";

$cookies.cook\_user\_email = "";

window.location = "index.html";

return;

}

else

{

return false;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* admin\_register \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$scope.user\_register = function()

{

$http.post('user\_register.php',{

'name':$scope.name,'email':$scope.email,'password':$scope.password,

'mobile': $scope.mobile,'field\_1': $scope.field\_1,'field\_2': $scope.field\_2,

'field\_3': $scope.field\_3,'field\_4': $scope.field\_4 })

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

alert("Registered successfully");

window.location = "user\_login.html";

return;

}

else if(data.success == 2)

{

alert("Please Fill All Fields");

}

else if(data.success == 0)

{

alert("Error");

}

else

{

alert(" Un Successfull");

}

});

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Admin Login \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// sign in button

$scope.admin\_login = function()

{

$http.post('admin\_login.php',

{'email': $scope.email, 'password':$scope.password})

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

alert("Login Successful");

$cookieStore.put("cook\_admin\_email",data.email);

window.location = "admin\_home.html"; // Home Page

return;

}

else if(data.success == 2)

{

alert("Please Fill All Fields");

}

else

{

alert("Login Unsuccessful");

}

});

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Cookies Login \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$scope.cook\_admin\_email = $cookieStore.get("cook\_admin\_email");

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* admin\_register \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$scope.admin\_register = function()

{

$http.post('admin\_register.php',{

'name':$scope.name,'email':$scope.email,

'password':$scope.password,'mobile': $scope.mobile})

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

alert("Registered successfully");

window.location = "admin\_login.html";

return;

}

else

{

alert("Invalid Inputs");

}

});

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Update Admin Info \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$http.post('get\_admin\_info.php')

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

$scope.details = data.details;

}

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Update User Info \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$http.post('get\_user\_info.php',

{

'email':$scope.cook\_user\_email

})

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

$scope.userdetails = data.details;

}

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* User Login \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$scope.myinfovar = true;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Admin Update Login \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

$scope.update\_info = function(email,password,name,mobile)

{

$scope.myinfovar = false;

$scope.email = email;

$scope.password = password;

$scope.name = name;

$scope.mobile = mobile;

//window.location = "home.html";

}

$scope.save\_info = function()

{

$http.post('admin\_update.php',{

'name':$scope.name,'email':$scope.email,

'password':$scope.password,'mobile': $scope.mobile})

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

alert("Submited successfully");

window.location = "admin\_post\_info.html";

return;

}

else

{

alert("Invalid Inputs");

}

});

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* User Login \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// sign in button

$scope.newpassword = function()

{

$http.post('newpassword.php',

{ 'email': $scope.email, 'password':$scope.password,

'field\_3': $scope.field\_3, 'field\_4':$scope.field\_4

})

.success(function(data, status, headers, config)

{

if(data.success == 1)

{

alert("Password Reset Successful");

window.location = "index.html"; // Home Page

return;

}

else if(data.success == 2)

{

alert("Please Fill All Fields");

}

else

{

alert("Login Unsuccessful");

}

});

}

});

**User\_login.php**

<?php

/\* Following code will match admin login credentials \*/

//user temp array

$response = array();

// include db connect class

require\_once \_\_DIR\_\_ . '/db\_connect.php';

// connecting to db

$db = new DB\_CONNECT();

// check for post data

$data = json\_decode(file\_get\_contents("php://input"));

$get\_empid = mysql\_real\_escape\_string($data->email);

$get\_password = mysql\_real\_escape\_string($data->password);

if(empty($get\_empid) || empty($get\_password))

{

$response["success"] = 2;

echo json\_encode($response);

}

else

{

$result = mysql\_query("SELECT \* FROM user\_login WHERE email = '$get\_empid' AND password = '$get\_password' ");

if (mysql\_num\_rows($result))

{

$Allresponse = mysql\_fetch\_array($result);

// temp user array

$response = array();

$response = $Allresponse;

$response["success"] = 1;

echo json\_encode($response);

}

else

{

// success

$response["success"] = 0;

// echoing JSON response

echo json\_encode($response);

}

}

?>

**Index.html**

<!DOCTYPE html>

<html ng-app="myapp">

<head>

<!-- Basic -->

<meta charset="utf-8">

<meta name="keywords" content="HTML5 Template" />

<meta name="description" >

<meta name="theme-color" content="#050e82" />

<!-- Mobile Metas -->

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- Web Fonts -->

<link href='css/css.css' rel='stylesheet' type='text/css'>

<link rel="stylesheet" href="fonts/font.css">

<!-- Vendor CSS -->

<link rel="stylesheet" href="vendor/fontawesome/css/font-awesome.css">

<link rel="stylesheet" href="vendor/owlcarousel/owl.carousel.css" media="screen">

<link rel="stylesheet" href="vendor/owlcarousel/owl.theme.css" media="screen">

<link href="vendor/owl-carousel/owl.transitions.html" rel="stylesheet" media="screen">

<link rel="stylesheet" href="vendor/flexslider/flexslider.css" media="screen">

<link rel="stylesheet" href="vendor/chosen/chosen.css" media="screen">

<link rel="stylesheet" href="vendor/magnific-popup/magnific-popup.css" media="screen">

<!-- Theme CSS -->

<link rel="stylesheet" href="css/theme.css">

<link rel="stylesheet" href="css/theme-animate.css">

<!-- Style Switcher-->

<!-- Head libs -->

<script src="vendor/modernizr/modernizr.js"></script>

<!--[if IE]>

<link rel="stylesheet" href="css/ie.css">

<![endif]-->

<!--[if lte IE 8]>

<script src="vendor/respond/respond.js"></script>

<script src="vendor/excanvas/excanvas.js"></script>

<![endif]-->

<style>

.container {

background-color: #050e82;

position:relative;

margin-top:100px;

}

.row{

margin-left:50px;

margin-right:50px;

text-align:center;

}

.login p{

position:relative;

text-align:center;

font-size:30px;

color:#ffffff;

}

p{

position:relative;

text-align:center;

font-family:''Quicksand'';

font-size:35px;

color:#ffffff;

font-weight:200;

}

h2{

position:relative;

text-align:center;

font-size:30px;

color:#ffffff;

font-weight:200;

}

h3{

text-align:center;

font-size:80px;

color:#ffffff;

font-weight:200;

}

img{

align:middle;

}

</style>

<!-- script back button -->

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

<script>

function onLoad()

{

document.addEventListener("deviceready", deviceReady, false);

}

function deviceReady()

{

document.addEventListener("backbutton", backButtonCallback, false);

}

function backButtonCallback()

{

navigator.app.exitApp();

}

</script>

<!-- angular js -->

<script src="js/angular-1.3.js"></script>

<script src="js/angular\_cookies.js"></script>

</head>

<body >

<div class="container">

<h2>Fuel Delivery On Demand</h2>

<div class="row">

<div class="col-md-12" style="margin-left:0px;" >

<img class="img-responsive" src="images/wall.jpg" width="400px" height="400px" >

</div>

<div class="login" style="margin-left:0px;">

<div class="col-md-4" >

<a href="user\_login.html"> <button style="background-color:white;font-weight:bold;color:#000;letter-spacing:1px">User Login</button> </a> <br><br>

<a href="user\_register.html"><button href="user\_register.html" style="background-color:white;font-weight:bold;color:#000;letter-spacing:1px">User Register</button> </a><br><br>

<a href="admin\_login.html"><button href="user\_register.html" style="background-color:white;font-weight:bold;color:#000;letter-spacing:1px">Admin Login</button> </a><br><br>

<a href="worker\_login.html"><button href="worker\_login.html" style="background-color:white;font-weight:bold;color:#000;letter-spacing:1px">Bunk Login</button> </a><br><br>

<a href="worker\_register.html"><button href="worker\_register.html" style="background-color:white;font-weight:bold;color:#000;letter-spacing:1px">Bunk Register</button> </a><br><br>

</div>

</div>

</div>

<script src="vendor/jquery/jquery.js"></script>

<script src="vendor/bootstrap/bootstrap.js"></script>

<script src="vendor/jquery.validation/jquery.validation.js"></script>

<script src="vendor/owlcarousel/owl.carousel.js"></script>

<script src="vendor/flexslider/jquery.flexslider-min.js"></script>

<script src="vendor/countdown/countdown.min.js"></script>

<script src="vendor/chosen/chosen.jquery.min.js"></script>

<script src="vendor/pricefilter/jquery.pricefilter.js"></script>

<script src="vendor/masonry/imagesloaded.pkgd.min.js"></script>

<script src="vendor/masonry/masonry.pkgd.min.js"></script>

<script src="vendor/uikit/uikit.js"></script>

<script src="vendor/magnific-popup/jquery.magnific-popup.js"></script>

<!-- Theme Base, Components and Settings -->

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

<!-- Style Switcher -->

<script type="text/javascript" src="style-switcher/js/switcher.js"></script>

<!-- angular js -->

<script src="js/angular\_product.js"></script>

<body onload="onLoad()">

</body>

</html>