PROJECT REPORT ON ONLINE MEDICINE INFO(MEDI-INFO)

by

RAMAN SRIVASTAVA

Dr. APJ Abdul Kalam Technical University

1805914928

Under Guidance of
Dr. PRAVEEN SACHAN

Submitted to the Department of Computer Applications in partial fulfillment of the requirements for the award of the degree

Master of Computer Applications (MCA) 2020

Lal Bahadur Shastri Institute of Management and Development Studies, Gaura Bagh, Kursi Road, Lucknow – 226026

CERTIFICATE

This is to certify that Mr./Ms. ____________Raman Srivastava___________

Roll No. ___1805314928______ a student of Master of Computer Application,

6th Semester of this institute has completed the industrial project (MCA671)

Titled “Online Medicine Information (Medi-Info) “in _______________________

of six months duration in partial fulfilment of Master of Computer Application degree program of LBSIMDS. To the best of my knowledge and belief, the above said report has been prepared by the above said student.

Head of Department

DECLARATION

I hereby declare that this_Project Report entitled “ONLINE MEDICINE INFO” and the work Described in it is my own work, and are carried out in accordance with the regulations of the A.P.J Abdul Kalam University. The work is original except, where indicated by special references in the text and no part of the project has been submitted for any other degree.

ACKNOWLEDGEMENT

It gives me immense pleasure in submitting the project report on “MEDI-INFO “for the partial fulfilment of the requirement for the degree of MCA.

First of all, I wish to express my sincere gratitude to my MCA faculty for allowing me to work on this report. I am also thankful to my college Lal Bahadur Shastri Institute of Management and Development Studies, its staff, who rendered their help during the period of my project work and provided me all privileges.

I am highly grateful to Mr. Praveen Sachan, for his encouragement and Guidance and valuable suggestion, which I incorporated in my report. I am also grateful to my respectful Teacher for helping me in the report finalization process and showing the confidence in me.

We are grateful to our project guide sir for the guidance, inspiration and constructive suggestion that help us in the preparation of this project. Last but not the least I wish to avail myself of this opportunity express a sense of gratitude and love to my friends and my beloved parents for their support, strength, and help for everything.

Thanking You

Raman Srivastava

MCA 6 (semester)

ABSTRACT

The Goal of any system development is to develop and implement the system cost efficiently. Computer Software succeeds-when it meets the needs of the people who use it, when it performs flawlessly over a long period of time, when it is easy to modify and even easier to use-it can and does change things for the better. But when the software fails-when its users are dissatisfied, when it is error prone, when it is difficult to change and even harder to use-bad things can and do happen.

We all want to build website that makes things better, avoiding the bad things that lurk in the shadow of failed efforts. To succeed we need discipline when software is designed and built. Many individuals and companies still develop software haphazardly, even as they build system to service the most advanced technologies of the day. As a result, the quality of the software that we produce suffers and bad things happen.

The MEDI-INFO can be entered using a user email and password. It is accessible by the registered patients or doctors. Only doctor can add data and patient categories into the database and the registered users can read the updated medicine. The data can be retrieved easily. The data are well protected and makes the data processing very fast.

This emphasis in this project is to document the important concepts and the techniques used for the successful development of this project.

I do hope fervently that, through this project the readers will get a real picture of what the project is all about. I also wish that my web application satisfies all the needs and requirements of the organization for which it is meant.

TABLE OF CONTENT

INTRODUCTION

Medi-Info will be a Web app as well as an Android app to help people with their medicine and doctors' routine. It will store all the medications prescribed him or her and help maintain the stock till the time needed.

Keeping track of when you took your last dose of a painkiller can be a pain. So, can trying to remember if you took that daily pill. Well, Medi-Info will make your life easier. Not only will it keep track of your medications, when you take them, when you need to renew them, and even when you can next take that "as needed" medication, it can also help you keep a health journal of all your health events. This might include sugar levels, weight, how you feel or anything else that might be relevant over the long term. Not only can this help you assess your long-term health, it can also be a great tool for your doctor.

DESCRIPTION

The Medi-Info application is meant to be used by anyone that is responsible for their own health. One specific use case where Medi-Info would shine is for a woman who is trying to get pregnant. She could track her cycles in the Health Diary. If she needs to take medications, Medi-Info can help her stay on top of ensuring that she stays consistent with taking her pills which is actually needed.

Another specific use case is for a person who is trying to lose weight. There may be some medications to track, there will be weight to be entered into the diary, and the Health Report will be useful when talking to his or her doctor.

One final use case (that I am specifying - there are a number of other possible use cases) is for the individual that is away from home when a health event occurs. They will have their recent health information at hand for the emergency room physician, including what medications they are taking. Once they are released, the Pharmacy Lookup tool will help them identify where the nearest pharmacy is so that they can fill their prescriptions and get back to their vacation with a minimum of fuss. They can even use the Health Data to let their regular doctor know about their health event.

OBJECTIVE

- “MEDI-INFO” is a website which is capable of displaying, medicine of different disease.
- Simplicity is one of the main characteristics of Medi-Info. It offers a very simple UI.
- Medi-Info purpose is to provide Patient proper routine and updatetation of the medicine, according to doctor’s prescription.

SCOPE AND FEATURES

Medi-Info has the following scope and features:

- **Pill Lookup -** Get more information on the medicine you are taking.
- **Recurring Event Tracking -** Tracks and reminds you of events such as taking your medicine (which also will let you know if you have taken that pill or not) or testing your blood sugar.
- **Countdown Tracking -** Tracks when the last time was that you took a medication that is "as needed" and lets you know when you can take it again.
- **Health Diary -** Lets you track your health history. Whether you are trying to get pregnant, tracking your blood sugar, or just looking to keep track of your health history, the diary is the place for your daily data. Your pill usage will be imported to the diary automatically.
- **Health Data -** Keep your health data in one place including doctor's contact information, health insurance information, pharmacy information, and any of those other bits of information that usually get lost in the shuffle of life.
- **Health Report -** Print out, email or just display a report of recent activities. This is most useful for health professionals or other individuals that are helping you.

PROJECT CATEGORY

Internet (Web Designing) including MEAN: --

The project is based on client server architecture. In client server architecture the interface for user is created in any programming language (Front end) and the database where data is stored is called back end. User interface is treated as client to which user request for data and user interface sends the user request to server or database and then server returns the data required by the client program manipulation.

All views that are theoretically updatable can be updated through the system. Views that are theoretically updatable can be updated through the system.

- The ability to create multiple relations among tables and store data in it.
- An interactive Validate API.
- Retrieval of information form as requirement from Data.

SYSTEM ANALYSIS

Systems analysis is the process of examining a business situation for the purpose of developing a system solution to a problem or devising improvements to such a situation. Before the development of any system can begin, a project proposal is prepared by the users of the potential system and/or by systems analysts and submitted to an appropriate managerial structure within the organization

PROJECT PROPOSAL: --

The project proposal is the attempt to respond to or take advantage of a particular situation and is an essential element for correctly launching the system analysis. Although there are no hard and fast rules as to the form and content of the project proposal, the proposal should address the following points:

- The specifics of the business situation or problem.
- The significance of the problem to the organization.
- Alternative solutions.
- The possible use of computer information systems to solve the problem.
- The various people interested in or possessing knowledge relevant to the problem.

System projects that are to be shared by a number of departments and users are usually approved by a committee rather than an individual. A project proposal is submitted to a committee that determines the merits of the proposal and decides whether or not to approve it. The committee is made up of people from various functional areas of the organization who have an interest in the operation and information of the proposed system.

IDENTIFICATION OF NEED

Requirement specification provides the developer and the Patient with the means to assess quality once software is built. The requirement analysis task is a process of discovery, refinement, modeling, and specification. Requirement analysis allows the developer to refine the system allocation and build models of the data, functional and behavioral domains that will be treated by system.

After detailed study of the identification of needs, following are the requirements of the project that includes:

- New system should not be too costly.
- New system should be fully automated.
- Implementation of the new system should be done in reasonable time.
- New system should be as simple as possible
- New system should provide new member registration
- New system should provide direct saving of the webpages in the user’s account.
- New system should provide a facility to keep reminders which can be received personally through e-mails
- New system should provide a means to store and edit personal notes.
- New system should provide sharing feature within the website.
- New system should also provide other features like search, edit, delete.

PRELIMINARY INVESTIGATION

The first phase of the systems development life cycle is preliminary investigation. Due to limited resources an organization can undertake only those projects that are critical to its mission, goals, and objectives. Therefore, the goal of preliminary investigation is simply to identify and select a project for development from among all the projects that are under consideration. Organizations may differ in how they identify and select projects for development. Some organizations have a formal planning process that is carried out by a steering committee or a task force made up of senior managers. Such a committee or task force identifies and assesses possible computer information systems projects that the organization should consider for development. Other organizations operate in an ad hoc fashion to identify and select potential projects. Regardless of the method used, and after all potential projects have been identified, only those projects with the greatest promise for the well-being of the organization, given available resources, are selected for development.

The objective of the systems-investigation phase is to answer the following questions: What is the business problem? Is it a problem or an opportunity? What are the major causes of the problem? Can the problem be solved by improving the current information system? Is a new information system needed? Is this a feasible information system solution to this problem?

The preliminary-investigation phase sets the stage for gathering information about the current problem and the existing information system. This information is then used in studying the feasibility of possible information systems solutions.

It is important to note that the source of the project has a great deal to do with its scope and content. For example, a project that is proposed by top management usually has a broad strategic focus. A steering committee proposal might have a focus that covers a cross-function of the organization. Projects advanced by an individual, a group of individuals, or a department may have a narrower focus.

A variety of criteria can be used within an organization for classifying and ranking potential projects. For planning purposes, the systems analyst—with the assistance of the stakeholders of the proposed project—collects information about the project. This information has a broad range and focuses on understanding the project size, costs, and potential benefits. This information is then analyzed and summarized in a document that is then used in conjunction with documents about other projects in order to review and compare all possible projects. Each of these possible projects is assessed using multiple criteria to determine feasibility.

FEASIBILITY STUDY

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The analyst conducting the study gathers information using a variety of methods, the most popular of which are:

- Interviewing users, employees, managers, and Patients.
- Observing or monitoring users of the current system to determine their needs as well as their satisfaction and dissatisfaction with the current system.
- Collecting, examining, and analyzing documents, reports, layouts, procedures, manuals, and any other documentation relating to the operations of the current system.
- Modeling, observing, and simulating the work activities of the current system.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

These components are:

1. Economic feasibility—

   The economic viability of the proposed system. The proposed project's costs and benefits are evaluated. Tangible costs include fixed and variable costs, while tangible benefits include cost savings, increased revenue, and increased profit. A project is approved only if it covers its cost in a given period of time. However, a project may be approved only on its intangible benefits such as those relating to government regulations, the image of the organization, or similar considerations.

2. Technical feasibility—

   The possibility that the organization has or can procure the necessary resources. This is demonstrated if the needed hardware and software are available in the marketplace or can be developed by the time of implementation.

3. Operational feasibility—

   The ability, desire, and willingness of the stakeholders to use, support, and operate the proposed computer information system. The stakeholders include management, employees, Patients, and suppliers. The stakeholders are interested in systems that are easy to operate, make few, if any, errors, produce the desired information, and fall within the objectives of the organization.

SOFTWARE REQUIREMENT SPECIFICATION

SRS is a document that completely describes what the proposed software should do without describing how the software will do it. The SRS is the medium through which the client and the user needs are accurately specified.

Advantages of SRS:

- An SRS establishes the basis for agreement between the client and the supplier on what the software product will do.
- An SRS provides a reference for validation of the final product
- A high-quality SRS is a prerequisite to a high-quality software.
- A high-quality SRS reduces the development cost.

Characteristics of an SRS:

A good SRS is:

1. Correct

2. Complete

3. Unambiguous

4. Verifiable

5. Consistent

6. Ranked for importance and/or stability

7. Modifiable

8. Traceable

Project’s SRS as follows:

Functional Requirement:

Functional requirements are those requirements which affect the system and make the system working. It describes what the system should do. Functional requirements of the system are-

- Registration of the Patient with unique ID and PASSWORD.
- Patient can change the password or mobile number any time, if required.
- Patient can search, read, delete, share, & access the category of their choice after login.
- patient can change our profile.

Doctoristrator section:

This section can be accessed by providing Doctor ID and password. In this section, the Doctor can view the user information. Doctor can view all the reviews and feedbacks sent to him by the patients.

User section:

Users can register and create their own profiles. After login users can perform different operations and access different services.

Non-Functional Requirement

Non-functional requirement describes how system should work or behave. Non-functional requirements of the system are-

Availability- The system should be 24/7 available. The Patients do not want restrictions on the times when they can access webpages or notes.

Reliability- The system should guarantee the reliability to run on any platform or machine and password protected.

Scalability- Since the project is planning to increase its Patient base, scalability should be assured by the system. The best efforts that can be made consist of enabling multiple Patients to access the website at the same time.

Usability- The users of the system are not computer experts, and hence particular attention should be paid to usability. Thus, the web interface should be clear, concise, and easy to use.

Flexibility- The system should be flexible enough to provide access to the webpage from any machine on any platform at any time.

Compatibility- The system should avoid compatibility issues with any system. The system should be compatible to run on any platform.

RESOURCES (HARDWARE & SOFTWARE)

Hardware Requirements:

- Server Side

Processor	Dual Core or above
RAM	4 GB
Disk Space	500 GB
Monitor	15-inch Color
Keyboard	108 Key Normal
Mouse	3 Button Normal Mouse

- Client Side

Processor	Dual Core or above
RAM	4 GB
Disk Space	500 GB
Monitor	15-inch Color
Keyboard	108 Key Normal
Mouse	3 Button Normal Mouse

Software Requirements:

- Server Side
  Internet Explorer or Google Chrome
  Windows XP or Above
  Visual Studio 2015
  ExpressJs, NodeJS
- Client Side
  Internet Explorer or Google Chrome
  Angular
- Windows XP or Above
- MS Office

TOOLS & PLATFORM

This project is a web application that is developed in Mean Stack using angular as Front-end tool and having express and node as server as a back-end tool.

MODERN WEB APPLICATION

The strategy in web development has changed dramatically in the recent years, thanks to the improvement of information technology. The browser is no longer used to serve static information. JavaScript is used heavily to serve dynamic elements in the browser. The browser is becoming a kind of manipulating system, which avails itself in the net of various data sources- the services of developer's servers (Jörg, 2016).

The modern web application, which is called web app, connects to the server to retrieve data dynamically. It only exists in the browser. When users approach the app for the first time, the app is rendered by the server and supports its services such as database access or transaction (Mozilla Inc, 2016). Deriving from modern web application, Medi-Info's internal project is designed to build application service provider called an APIs (Application Programming Interfaces) and use JSON (JavaScript Object Notation) as formatting language.

Atwood (2017), as a collar to Tim Bernes Lee's Rule of Least Power, proposed Atwood's Lay which states that any application that can be written in JavaScript, will eventually be written in JavaScript. Ryan Dahl (2009), wrote Node.js, an open-source, cross-platform JavaScript runtime environment. After several years of enhancement, it has combined with Google's V8 JavaScript engine, an event loop and low-level I/O API, and has supported executing JavaScript code in the server side.

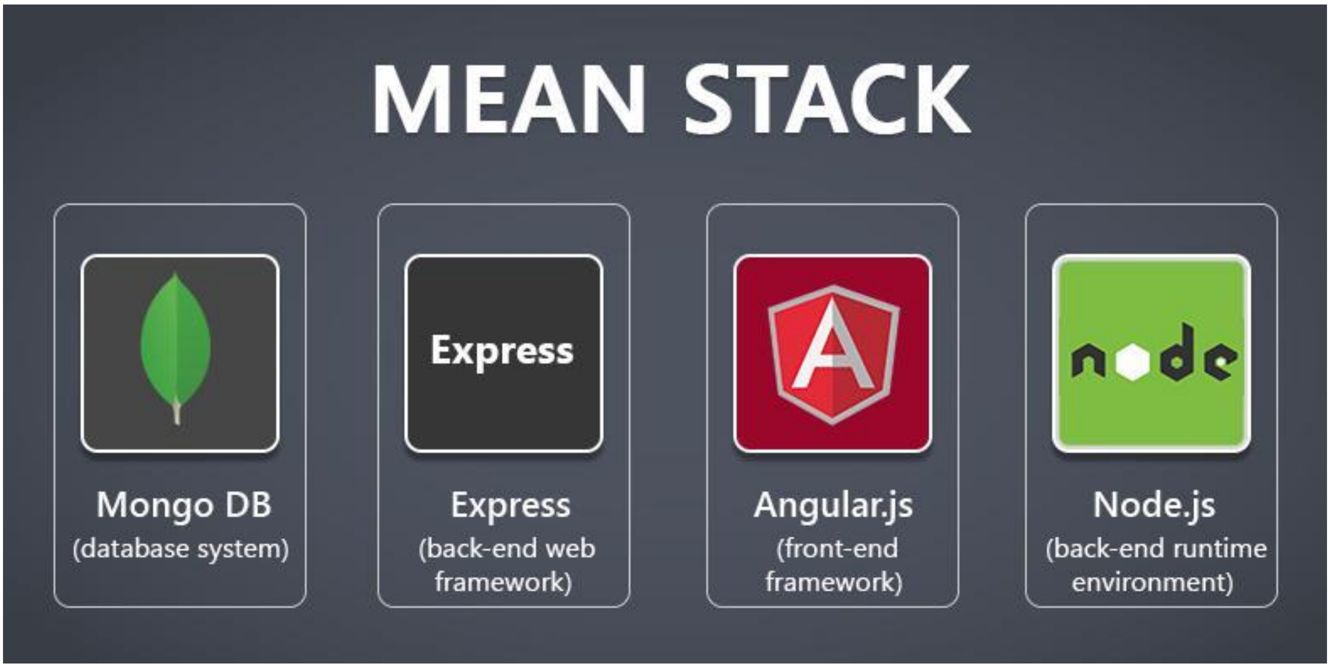
Single Page Application

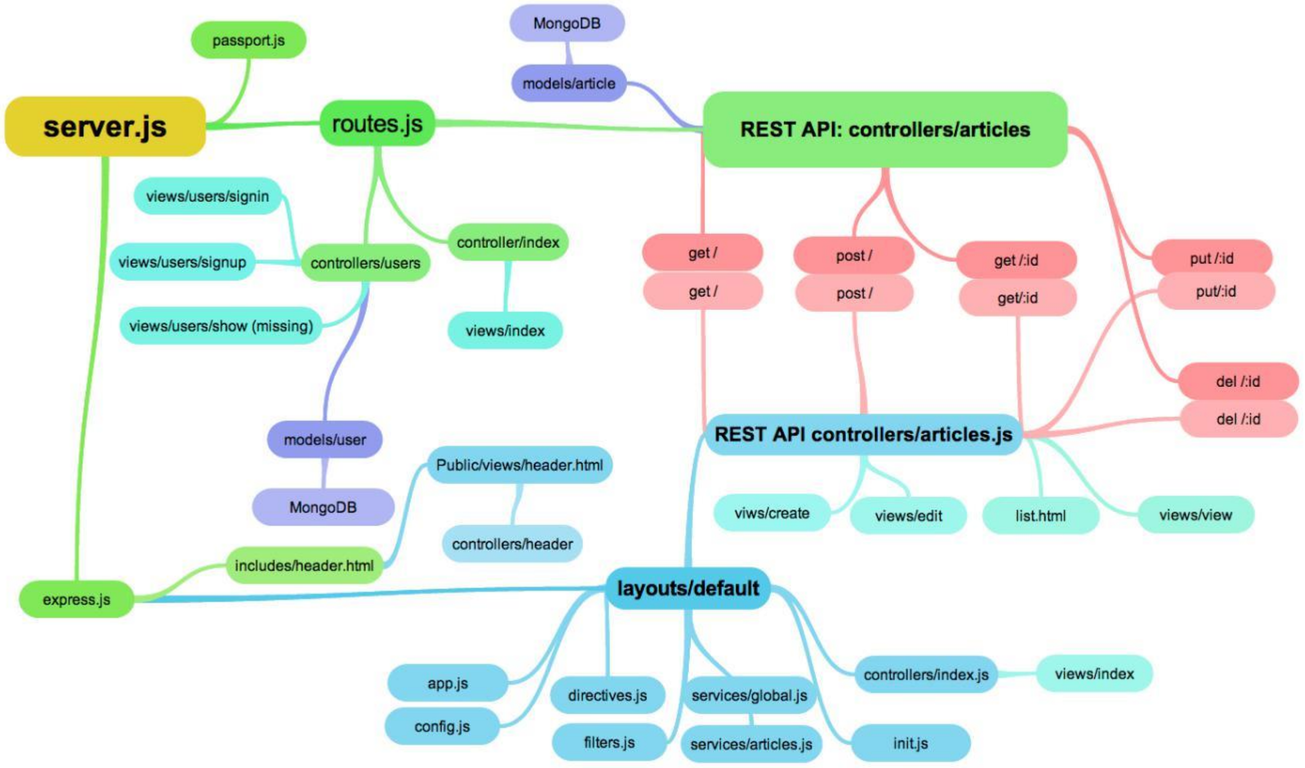
Instead of using numbers of HTML pages to display, SPA is a web application or website that is embedded in a single HTML page. The main goal of SPA is providing a more fluid user experience and richer interface. There is no limitation for the SPA. It can be a small application which simply provides CREATE, UPDATE, DELETE service, like a to-do list. SPA can scale into much more complex level with large amounts of requests. A Single Page Application contains countless libraries, templates and scripts in many folders depending on its complexity (Code School LLC, 2015).

The advantage of choosing this type of web application is to be able to update parts of a web site without sending request and reloading a full page. This feature is believed to be the most interesting feature of SPA. Nowadays, more developers are starting to approach SPA. It is much easier to develop and it is the best suit for RESTful API. From the start, everything is coded from the front-end of a web application using Ajax (to interacting with the server), HTML templates, a good MVC/MVVM framework, and of courses, an enormous amount of JavaScript (Wasson, 2013).

The MEAN Stack

In a full stack web application, JavaScript is the consistent language and syntax. The MEAN stack is comprised of four main JavaScript oriented technologies. MongoDB is the database, Express is back-end web framework, AngularJS is front-end web framework and Node.js is the back-end runtime environment. The reason why the stack is selected because developers will have improved view of greater picture by understanding the different areas and how they fit together. Frameworks and libraries are no longer limited in supporting user interfaces but also can be combined with other layers of applications. However, wrong architecture, low-level design and unbalance foundation in initial construction can dramatically affect the development process (Linnovate, 2014). Graph 1 explains the information of the MEAN Stack.

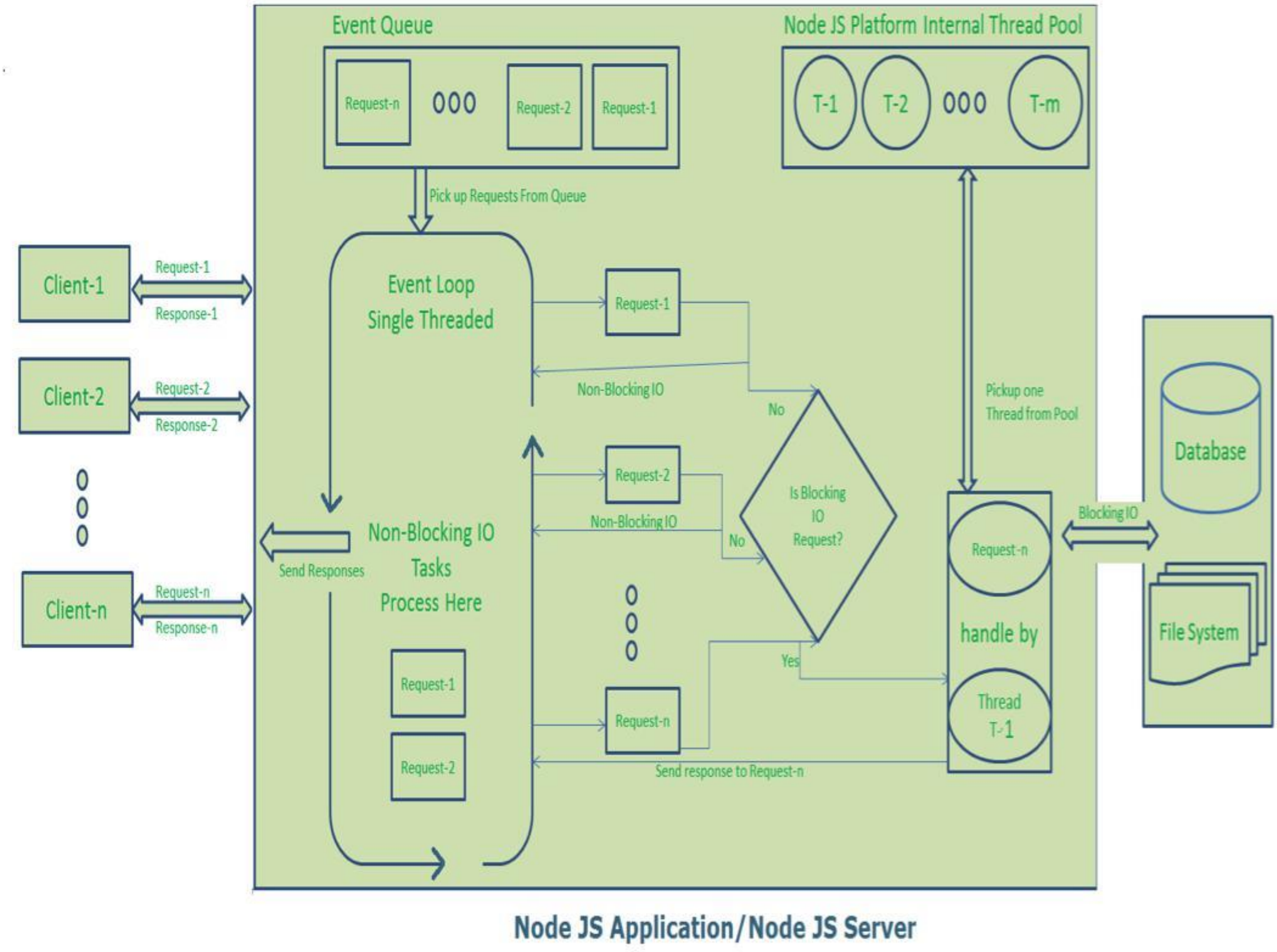




Node.js is a software platform allowing developers to create web servers and build web applications on it. Meanwhile PHP needs to run separate web server program such as Apache or Internet Information Services, Node.js server can be configured when building an application. However, it is not limited in web projects. More phone apps and desktop applications are built on top of Node.js. (Node.js Foundation, 2012).

Node.js provides many utilities' packages in order to create a powerful web server that runs JavaScript on the server side, for example, asynchronous I/O, single-threaded or multi-threaded, sockets, and HTTP connections. When building web applications using other server-side programming languages, such as Java, PHP, and .NET, the server creates a new thread for each new connection. Assumed that there is a popular website receiving more than two million requests daily, developers probably need more servers or even to invest in more hardware. This approach works well as long as the server has enough hardware infrastructure to provide services to the Patients. When the number of clients exceed the server's ability, it starts to slow down and the Patients have to wait (Holmes, 2015).

The single threaded, non-blocking I/O changes how a request is handled by the server. Instead of creating a new thread for each connection, the web server receives client requests and places them in a queue, which is known as Event Queue. The Event Loop picks up one client request from the queue and starts process it. If the request does not include any blocking I/O operation, the server process everything, creates a response and sends it back to the client. In the case that the request can block the I/O operation, there is a different approach. An available thread from the thread pool will be picked up and assigned to the client request. That thread has then processed the blocking I/O, processed the request and created a response to send back to the client (Jörg, 2016). Node servers can support tens of thousands of simultaneous connections. It does not allocate one thread per connection model, but uses a model process per connection, creating only the memory that is required for each connection (Monteiro 2014, 32). Graph 3 explains single threaded application.



- **Express**

  Even though Node.js is a great choice to make a web server, it is just a run time environment with many built-in modules. The default Node API is not able to handle complex applications as route management. A web framework is needed to do the heavy work. Express is a mature and flexible web framework to build web applications on top of the Node eco system. By default, the Express framework uses the Pug engine for supporting templates (Node.js Foundation, 2010).

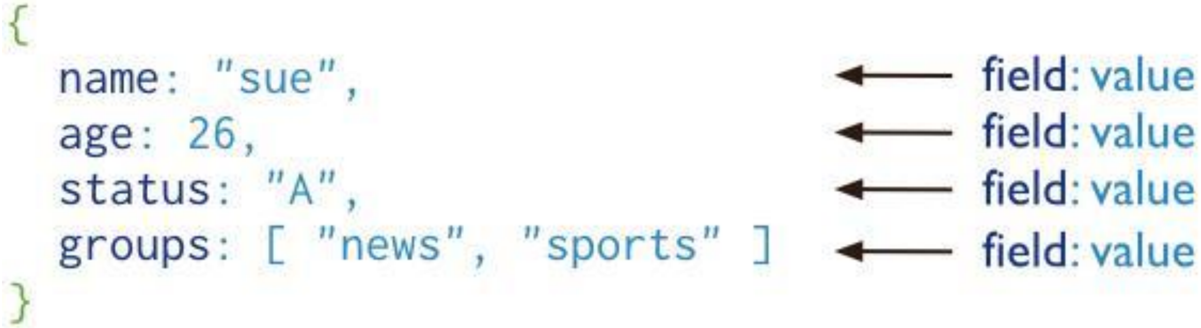
  Furthermore, Express reduces difficulties in setting up a webserver to handle incoming requests and return relevant response. That is the reason why it is the best choice to render HTML on the server side for large-scale application. It still provides an elegant set of features to deal with including SPA, the RESTful API, and the MEAN stack. Express follows good development practices, an oriented RESTful architecture that uses the main methods of the HTTP protocol (such as GET, POST, PUT, and DELETE), and very difficult-to-build complex web applications (Monteiro, 2014, 39).

- **AngularJS**

  At the moment, there is a enormous amount of front end libraries and frameworks supporting front end web development. AngularJS is one of them, which is designed to work with data directly in the front end. It allows developers to use HTML as template language and extend HTML’s syntax. There are two core features: Data binding and Dependency injection. AngularJS reduces most of codes, which are needed to write. Developers can use Node.js, Express, MongoDB to build a data-drive web application and pass data via HTTP protocols. Angular works with data directly in the frontend and puts the HTML templates together based on the provided data. Developers need less resources with this approach (Google, 2010).

- **MongoDB**

MongoDB is one of the most popular open source NoSQL databases written by C++. In 02/2015, MongoDB ranked at 4th in list of common databases. MongoDB is document-oriented, and it stores data in the key-value format, using binary JSON (BSON). It is agile, scalable and high performance. Node.js and MongoDB unleashed all the power for high-performance web applications with Express (MongoDB Inc, 2008).



```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

Graph 4 MongoDB data scheme (MongoDB Inc, 2008)

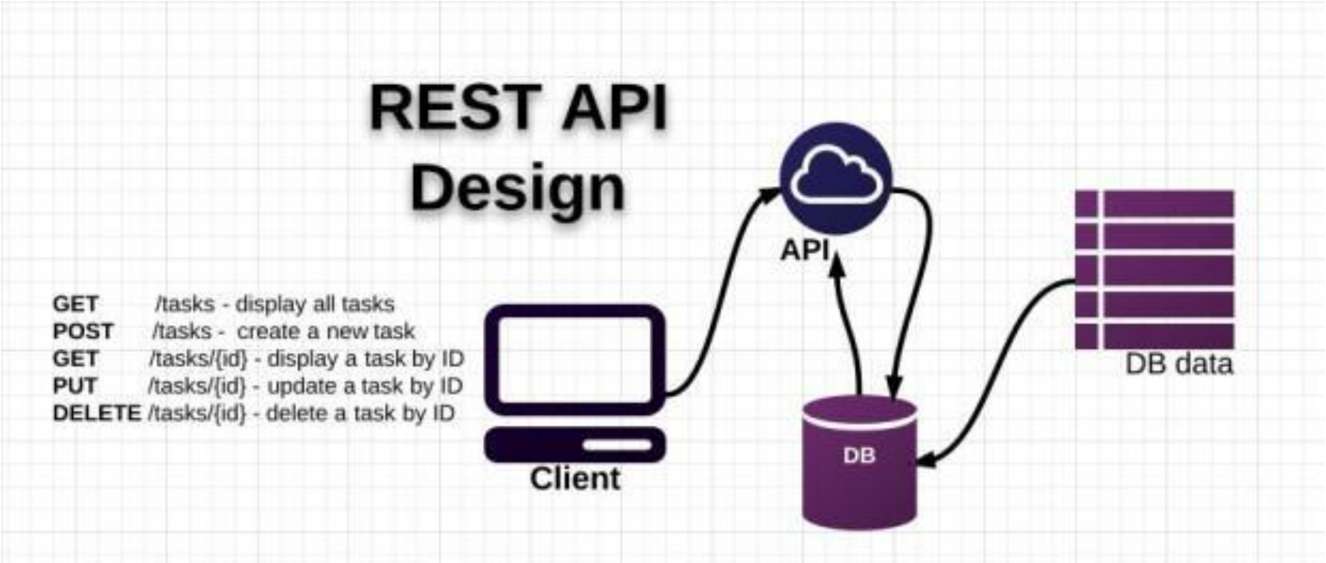
Document in MongoDB has the same structure as JSON, which means developers can map data as a key-value. We can understand the document as a record in MYSQL. Because MongoDB is written in C++, it is able to make a high-speed calculation, unlike other database management systems. Instead of writing a massive amount of SQL commands, MongoDB supports fairly complex and powerful queries in most use cases (MongoDB Inc, 2008).

REST API and JSON

In legacy applications, XML-RPC is the method to transfer data via HTTP protocols. It is extremely complicated and uses a large number of bytes to encode objects. However, REST API that uses JSON become the universal connector for data on the internet. The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language (W3Schools, 2013).

REST API

REST stands for Representation State Transfer, which is a stateless connection between clients and servers. It is modern client-server architecture using the HTTP protocol to communicate and JSON a formatting language. In a simple CRUD application as an example, REST API is used to create state-less interfaces for POST (create), PUT (update), GET (retrieve) and DELETE activities. The graph be-low shows the example of typical REST API design (Hunter II, 2013).


REST API
Design
GET /tasks - display all tasks
POST /tasks - create a new task
GET /tasks/{id} - display a task by ID
PUT /tasks/{id} - update a task by ID
DELETE /tasks/{id} - delete a task by ID
Client
API
DB
DB data

Graph 5 REST API design (Hunter II, 2013)

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. No matter what knowledge of JavaScript developer has, it is easy to read and write. There is no problem for machines to parse and generate. Compared to XML, it is JavaScript-friendly. It also has the advantage of being generally easier to write by hand than XML. JSON has better JavaScript support and is much simpler to write (Ecma-International, 2001).

Content Management System

Content is information produced through the editorial process and ultimately intended for human utilization via a publication. A CMS is a server-based, multiuser software that provides several effective content managements automatically. Editors can create new content, edit an existing one, or perform changes in content. Content manager will see differences right after commitment (Barker, 2016).

PLANING PHASE

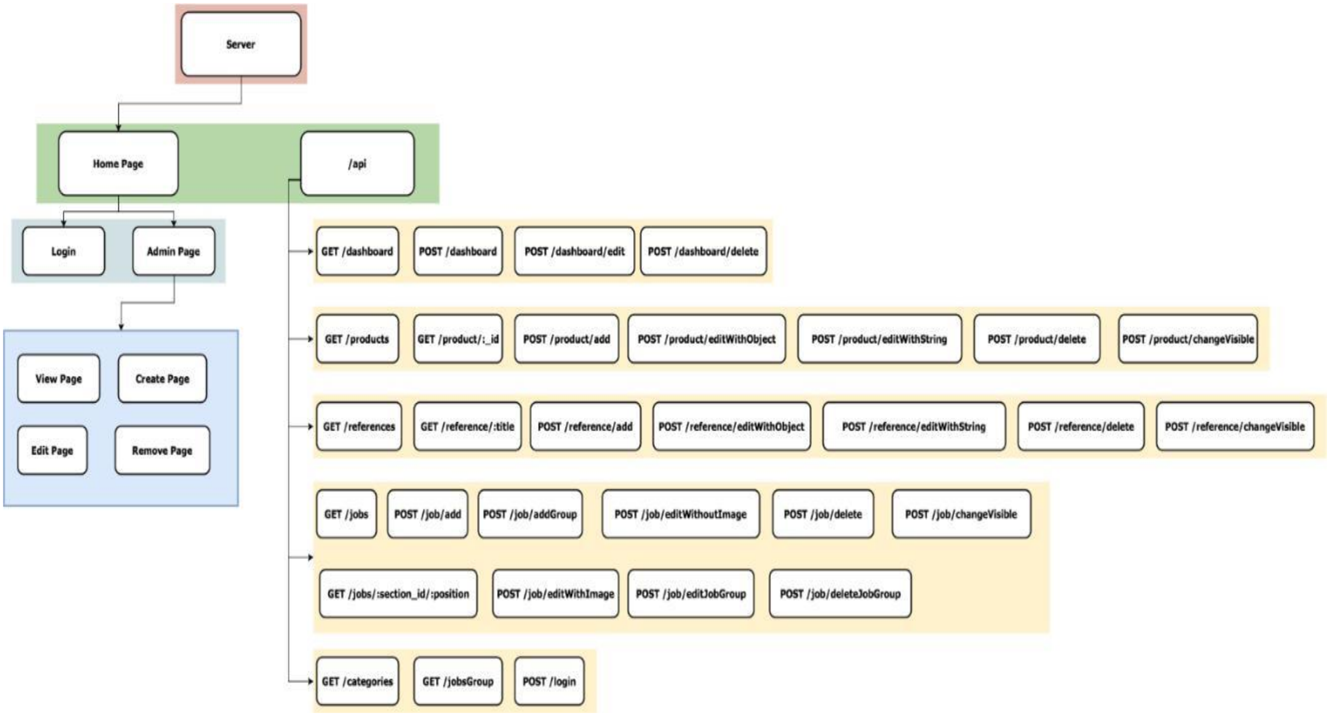
The problem came from after the training period. The developer who created the company's website left long time ago and it was inconvenient whenever he wanted to change the content of the website. It would be a grammar mistake, updating the image or adding new content. Hence, a task was an internal project before working with real clients, which is creating a simple content management system.

Goals and Target Users

The main goal of the project was helping the CEO managing the website by building a web application. The website's requirements were creating a web application and using MEAN stack as mandatory for practicing purpose. The layout of the website was asked to be simple and modern enough to use. JSON was the main formatting language of the API and other developers could connect to the CMS via REST API. The target users were Mark Dencket and Aaron Morley (COO and Project Manager). It was a tough and important challenge because there was only one developer responsible for the whole project. On the optimistic way, it was a chance to understand and learn how to make a complete website from scratch. Because there was only one developer in the team, tasks were required to be made on both server side and client side.

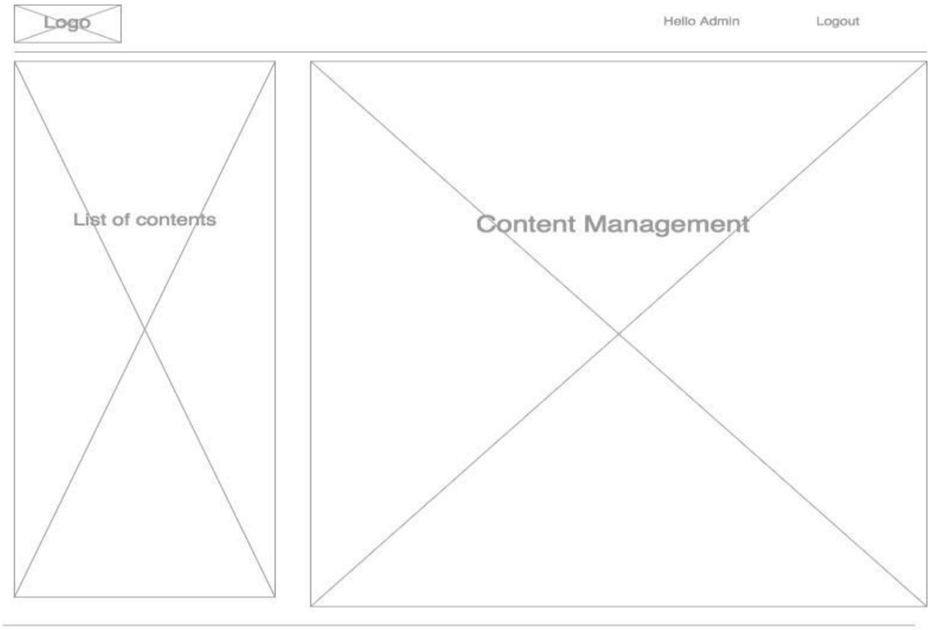
Site map and Wire frame Creation

At this point of development process, sample data was created, which could give a Patient an opportunity to judge how the entire site would look like. Based on the information gathered at the first stage, the sitemap was created. The graph below is the sitemap of the CMS website.



Graph 6 The site map of the CMS Application

The sitemap describes how the CMS was designed. From the view of clients using the web application, there are two main pages; log in and Doctor Page. The Doctor Page contain four main sub-pages for each content needed to be managed. Each sub-page corresponds to the CRUD elements. From the view of the developers who need to work with the API the site map displays URL end point and METHOD for each content of the company's website. However, the site map allows clients to understand what the inner structure of the project looks like. It does not give any hints about the user interface. A wire frame is created to provide a visual presentation of the user interface. It only describes the position of elements that will be added to the pages. Graph 7 displays the wire frame of CMS.



Graph 7 The wire frame of CMS

INSTALLATION

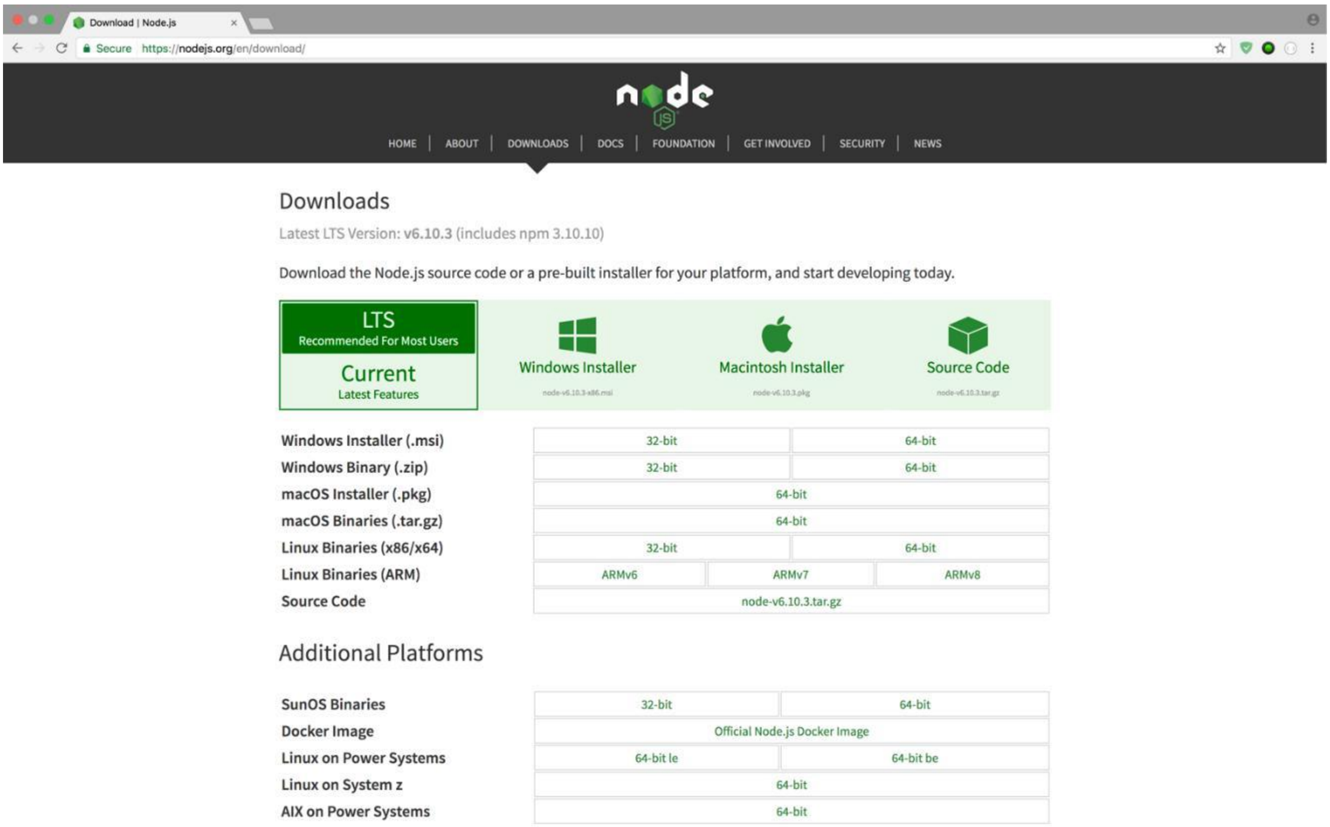
Before moving forward, Node.js, MongoDB and other necessary modules need to be installed. There must be a stable internet connection during installation and development process. MacOS is the main operating system and other Linux distros can follow the installation guide. Windows users can easily find a guide on the internet.

Node.js

As the main purpose is coding the web application on the Node.js runtime environment, it is crucial to install Node.js. Furthermore, npm is a tool to install modules. npm makes it easy for JavaScript developers to share the code that they have created to solve particular problems, and for other developers to reuse that code in their own applications (npm Inc, 2010).

Installing with GUI

Graph 8 displays the official page to download Node.js installation package. Depending on the local OS, users can download the necessary one. Developers should choose the most stable version in the LTS section. In the current tab, the page provides latest features for any operating system. However, they are not fully supported and can contain potential bugs. Downloading the most released version is not a wise move unless developers have been working for several years (npm Inc, 2010)



Downloads

Latest LTS Version: **v6.10.3** (includes npm 3.10.10)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

Windows Installer (.msi)	32-bit	64-bit	
Windows Binary (.zip)	32-bit	64-bit	
macOS Installer (.pkg)	64-bit		
macOS Binaries (.tar.gz)	64-bit		
Linux Binaries (x86/x64)	32-bit	64-bit	
Linux Binaries (ARM)	ARMv6	ARMv7	ARMv8
Source Code	node-v6.10.3.tar.gz		

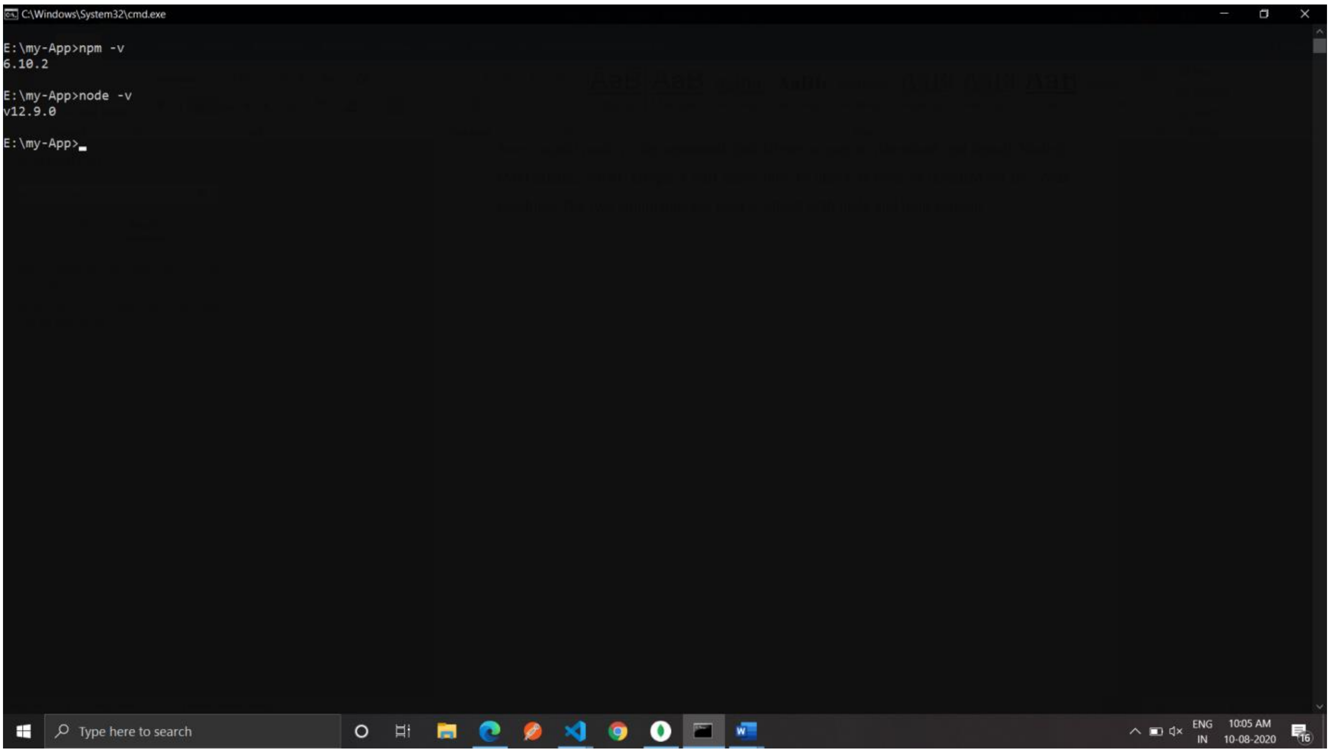
Additional Platforms

SunOS Binaries	32-bit	64-bit
Docker Image	Official Node.js Docker Image	
Linux on Power Systems	64-bit le	64-bit be
Linux on System z	64-bit	
AIX on Power Systems	64-bit	

Graph 8 Node.js download page (Node.js Foundation, 2012

Using terminal

Homebrew, the package manager on MacOS, is needed to install Node.js. It is easy to find how to install Homebrew at the official web site. When Homebrew is downloaded, *brew install node* is the command that allows a user to download and install Node.js (McFarland, 2014). Graph 9 will show how to check if node is installed on the local machine. The two commands are used to check both node and npm version.



```
E:\my-App>npm -v
6.10.2

E:\my-App>node -v
v12.9.0

E:\my-App>
```

Graph 9 Checking Node.js on local machine

MongoDB

Following the demonstration with Node.js, MongoDB can be installed using GUI application or terminal commands. https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/ displays the best way for favorite choice. After MongoDB is extracted, the following command shows how to connect to the mongod using command line. *mongod --dbpath pathToStoringDataDirectory* is the command to start MongoDB in a local machine. The pathToStoringDataDirectory indicates where MongoDB can store a database in a directory. (MongoDB Inc., 2008). If MongoDB is installed correctly, Graph 10 will show the result and connection port to MongoDB on local machine.

Utility Modules

This is the last stage of the installation process. There are three main modules which are extremely useful during the development process and can be used as command line. *Bower* is another package manager for front end development. *Nodemon* monitors any changes in source and restarts the server. *Express-generator* quickly creates an application skeleton. At this point, everything is ready to create a powerful MEAN stack application. *npm install -g express-generator bower nodemon* is the command, which shows how to use npm to install node modules globally. After the command is executed, utility modules can be invoked as command lines.

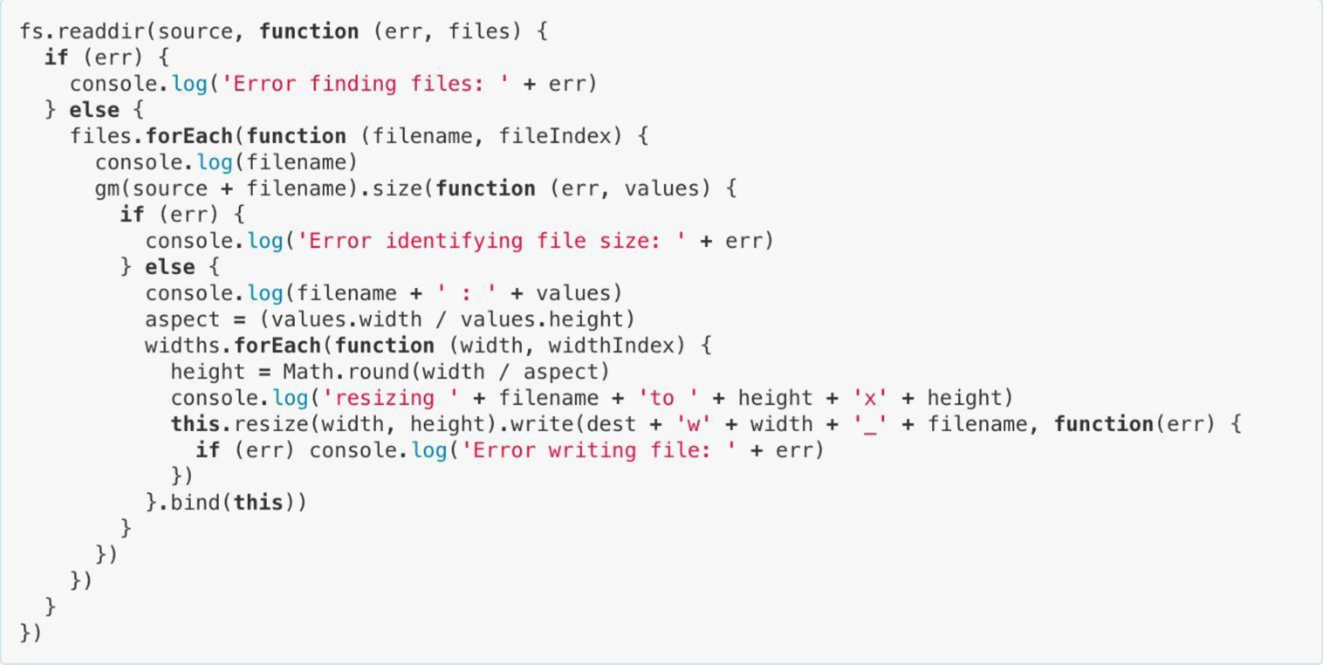
Mongoose

When MongoDB is running and providing a port to connect as shown in Graph 10, developer can open the database and simply work with it by running the *mongo* command. It is the same as connecting to MySQL with terminal command line. A developer can create a new table and modify a data. Graph 11 shows work with MongoDB in a terminal on a local machine.

However, creating and removing data manually is not preferred at all. It takes time and contains syntax errors easily. That is why Mongoose Module is preferred. Mongoose provides a straightforward, schema-based solution to model an application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box (Automatic, 2011).

Async and Q

One of JavaScript's core features is a function as a first-class object. They can be stored in variables, passed as arguments to functions, created within functions, and returned from functions (Cunningham, 2014). According to Cunningham, passed functions can be executed inside a function. This feature is called callback. Developers sometimes encounter numerous levels of callback functions which looks like Graph 12.



```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

Graph 12 Callback example (Ogden, 2012)

At the end of Graph 12, there are some}) declaring the ending of the nested functions. The deeper level of callback, the more difficult developers can find the final ending of chained executions. This low-level design is called a callback hell. Among callback hell solutions, Async and Q modules are chosen due to personal favor and clear documentation.

Async is a utility module which provides straightforward, powerful functions for working with asynchronous JavaScript. Although originally designed for use with Node.js and installable via *npm install –save async*, it can also be used directly in the browser (McMahon, 2014). Among around seventy functions provided by the module, parallel and series are two most used functions during the development process because they are suitable for the specific cases.

The other way to avoid callback function is using a promise object. It is one of the ECMAScript 2015 (6th Edition, ECMA-262) built-in support. If a function cannot return a value or throw an exception without blocking, it can return a promise instead. A promise is an object that represents the return value or the throw an exception that the function may eventually provide (Kowal, 2009). Kowal also wrote Q, a promise library for JavaScript. Compared to default promise object, Kowal's library provides more documentations, API and resources to follow.

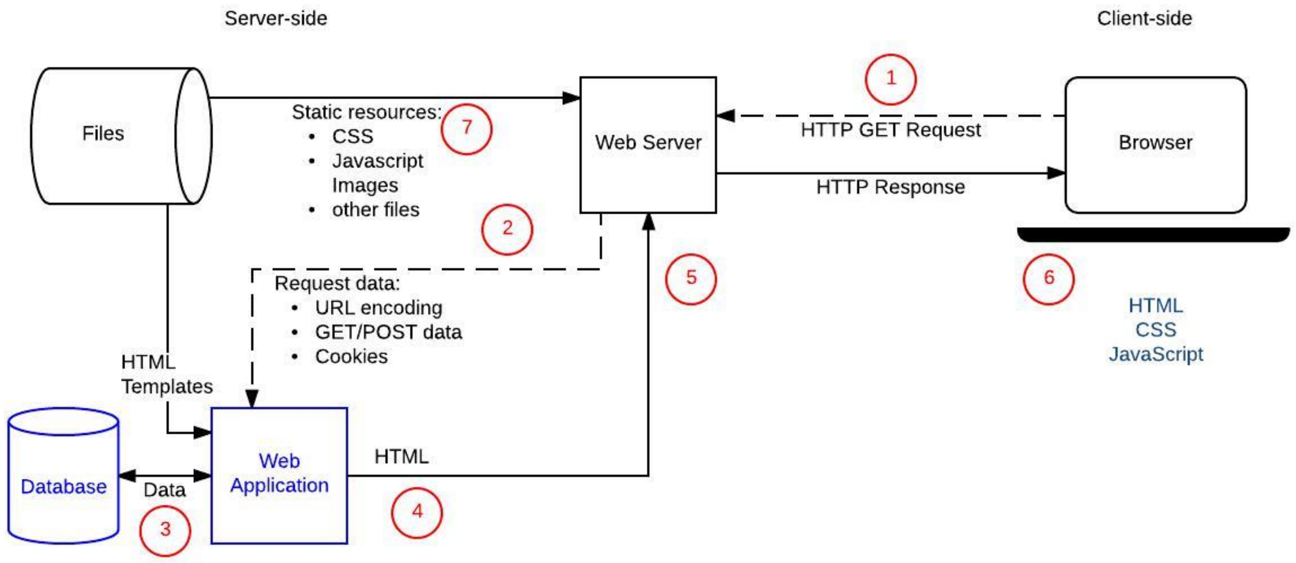
IMPLEMENTATION

For an implementation process to be successful, many tasks between different departments need to be accomplished in sequence. Companies strive to use proven methodologies and enlist professional help to guide them through the implementation of a system, but the failure of many implementation processes often comes from the lack of accurate planning in the beginning stages of the project due to inadequate resources or unforeseen problems that arise (Rouse, 2015).

The process of developing the application is divided into two main parts; server-side development and client-side construction. The goals of the server are rendering the main single HTML file and providing RESTful API. Meanwhile, the task of the front-end side is displaying the information and CRUD inter-face to manage the content. In personal opinion, the whole project is not difficult, but it contains an enormous amount of jobs to do before deploying.

Application Architecture

In this project, a browser retrieves data using the Hyper Text Transport Protocol (HTTP) to the web server. Other events happen in the browser (e.g. form submitted), an HTTP request is sent from the browser to the target server. The defined request handlers in receiving client's request messages, process them and answer the web browser with an HTTP response message. Clients get the status code and message indicating whether or not the request succeeded (Mozilla Inc, 2016).



SOFTWARE ENGINEERING PARADIGM

Software engineering is a layered technology. The foundation for software engineering is the process layer. Software engineering processes the glue that holds the technology layers together and enables ratios and timely development of computer software. Process defines a framework for a set of key process areas that must be established for effective delivery of software engineering technology.

Software engineering methods provide the technical how-toes for building software. Methods encompass a broad array of tasks that include requirements analysis, design, program construction, testing and support.

Software engineering tools provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another tool, a system for the support of software development, called computer-aided software engineering is established.

Modular approach is used for developing the proposed system. A system is considered modular if it consists of discrete components so that each component can be implemented separately and a change to one component has minimal impact on other components. Every system is a hierarchy of components. This system is not an exception. To design such hierarchies there are two approaches:

(1) Top down

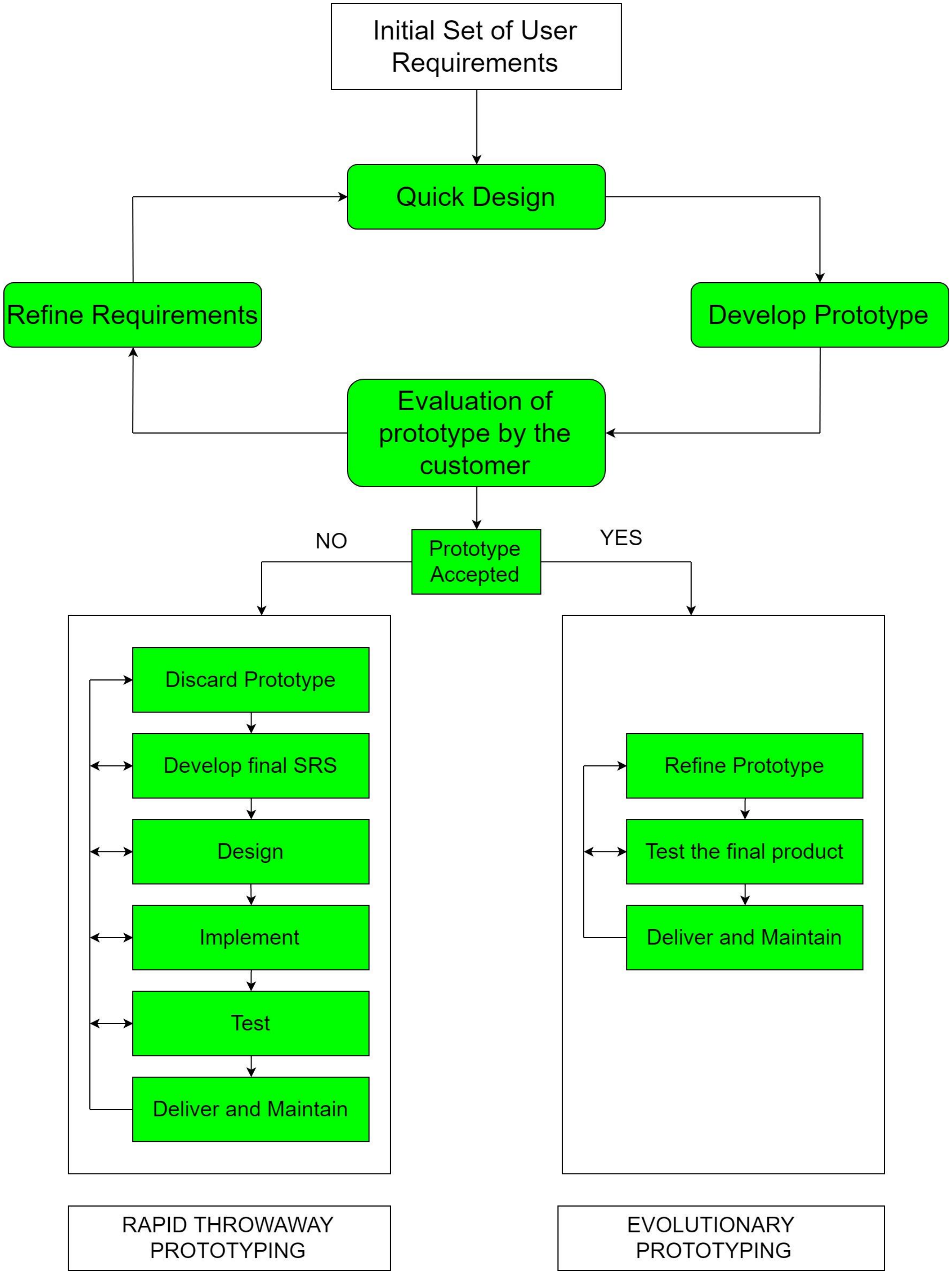
(2) Bottom up

Both approaches have some merits and demerits. For this system top down approach has been used. It starts by identifying the major components of the system, decomposing them into their lower level components and iterating until the derived level of detail is achieved.

Top down design methods often result in some form of stepwise refinement. Starting from an abstract design, in each step the design is refined to a more concrete level, until we reach a level where no more refinement is needed and the design can be implemented directly. A top down approach is suitable only if the specifications of the system are clearly known and the system development is from scratch. A bottom up approach starts with designing the most basic or primitive components and proceeds to higher level components that use these lower level components.

THE PROTOTYPE MODEL

The basic idea of prototyping is that instead of freezing the requirements before any design or coding can proceed, a throwaway prototype is built to help understand the requirements. This prototype is developed based on the prototype obviously undergoes design, coding & testing, but each of these phases is not done very formally or thoroughly. By using this prototype, the client can get an actual feel of the system, because the interactions with the prototype can enable the client to better understand the requirements of the desired system.


Initial Set of User Requirements
Quick Design
Refine Requirements
Develop Prototype
Evaluation of prototype by the customer
NO
Prototype Accepted
YES
Discard Prototype
Develop final SRS
Design
Implement
Test
Deliver and Maintain
Refine Prototype
Test the final product
Deliver and Maintain
RAPID THROWAWAY PROTOTYPING
EVOLUTIONARY PROTOTYPING

REASONS BEHIND USING PROTOTYPE MODEL

Since there is no existing system (computerized), prototyping model is an attractive idea.

- It helps to reduce the cost and time.
- It provides an early detection of errors.
- It allows the developers to have a greater control over the problem.
- It is an effective method of demonstrating the feasibility of a certain approach.
- Exposes developers to potential future system enhancements.

DATA MODEL

A data model is an abstract model that describes how data is represented and accessed. The term data model has two generally accepted meanings:

1. A data model theory, i.e. a formal description of how data may be structured and accessed.

2. A data model instance, i.e. applying a data model theory to create a practical data model instance for some particular application.

Data Model Theory

A data model theory has three main components:

- The structural part: a collection of data structures which are used to create databases representing the entities or objects modelled by the database.
- The integrity part: a collection of rules governing the constraints placed on these data structures to ensure structural integrity.
- The manipulation part: a collection of operators which can be applied to the data structures, to update and query the data contained in the database.

Data Model Instance

- A Data Model Instance is created by applying a Data Model Theory. This is typically done to solve some business enterprise requirement. Business requirements are normally captured by a semantic logical data model .This is transformed into a physical Data Model Instance from which is generated a physical database. For example, a Data modeler may use a data modeling tool to create an Entity-relationship model of the corporate data repository of some business enterprise. This model is transformed into a relational model, which in turn generates a relational database.

DATA FLOW DIAGRAM (DFD)

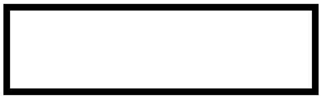
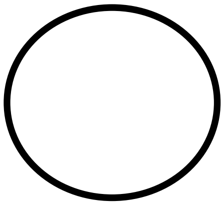
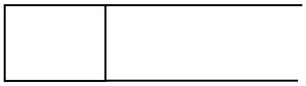
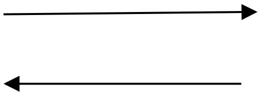
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

Types of DFD

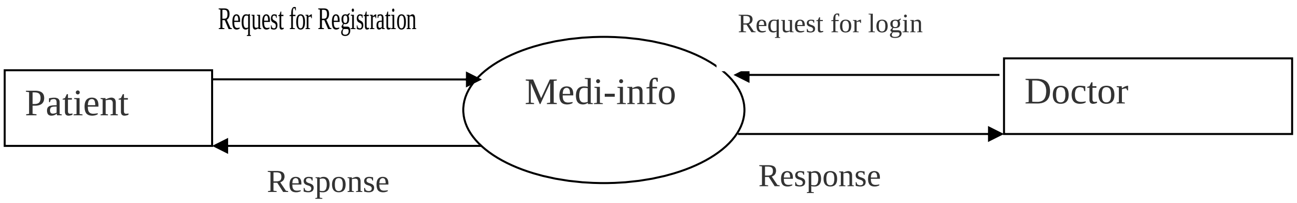
Logical DFD - This type of DFD concentrates on the system process and flow of data in the system.

Physical DFD - This type of DFD shows how the data flow is actually implemented in the system.

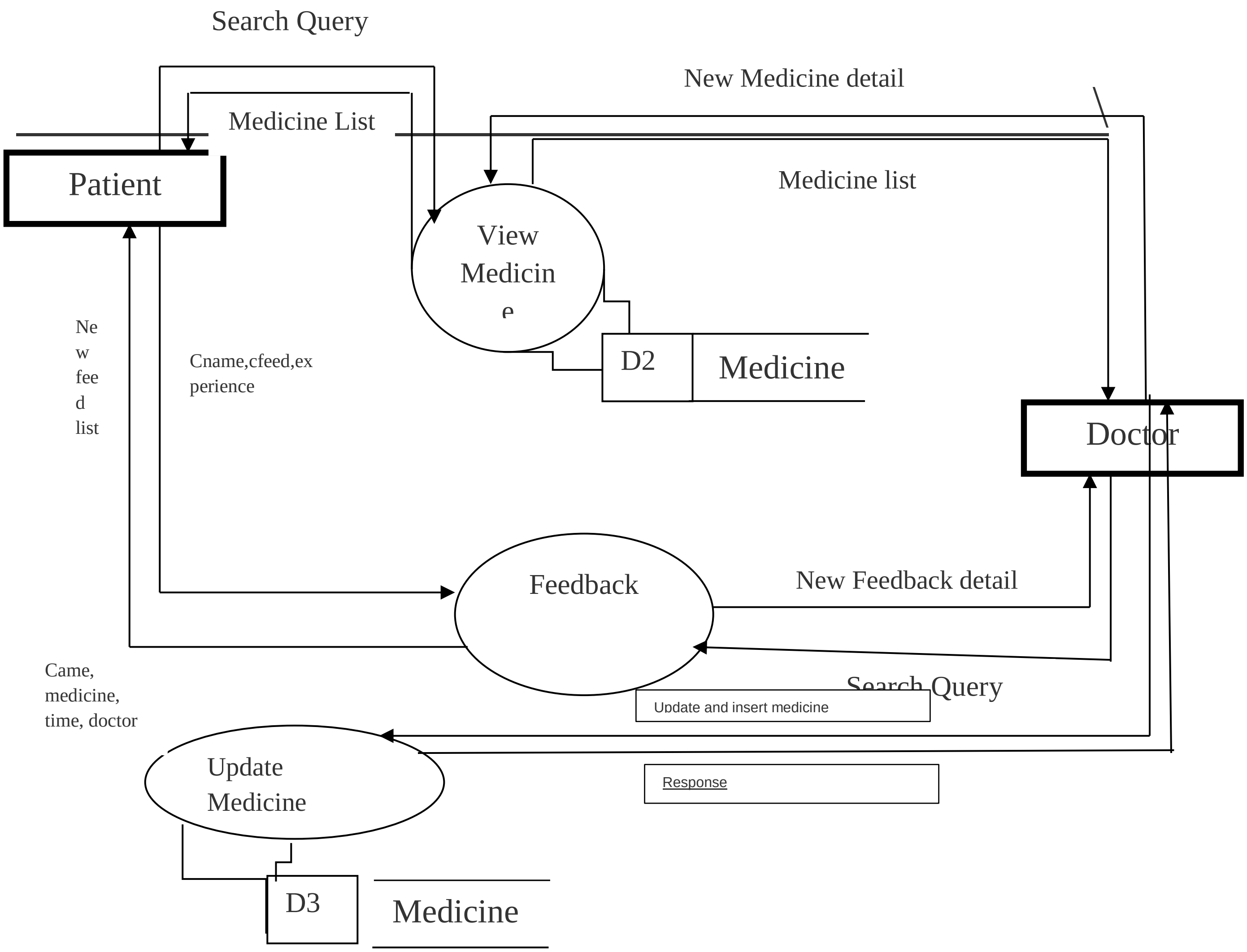
DFD Components

NAME	SYMBOL	DESCRIPTION
Entities		Entities are source and destination of information data. Entities are represented by a rectangle with their respective names.
Process		Activities and action taken on the data are represented by Circle or Round-edged rectangles.
Data Storage		There are two variants of data storage it can either berepresented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
Data Flow		Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

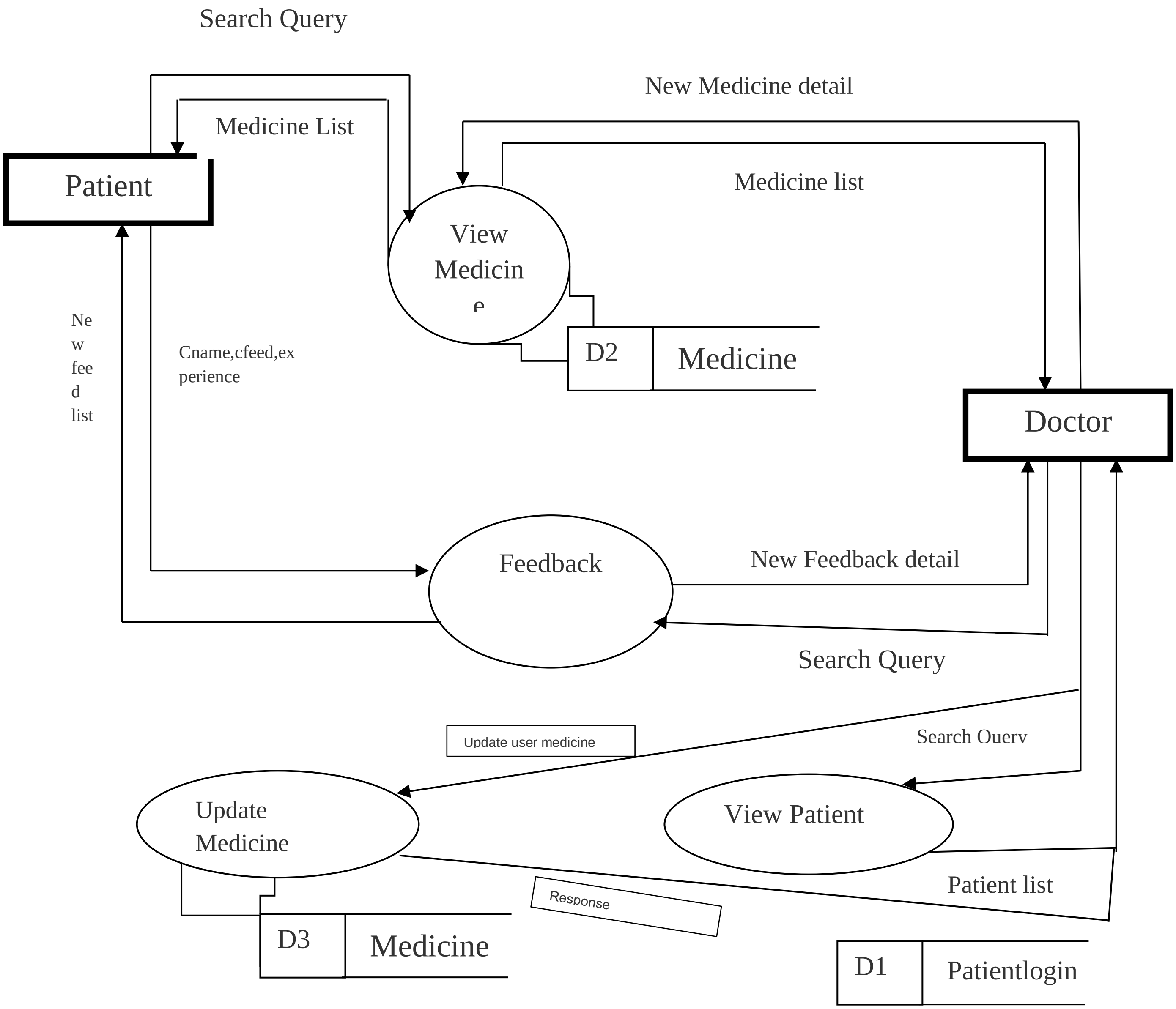
LEVEL 0 DFD: --



LEVEL 1 DFD: --



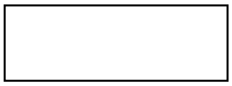
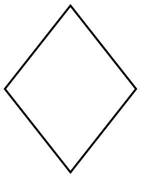
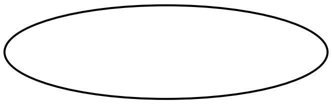
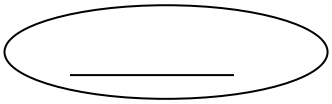
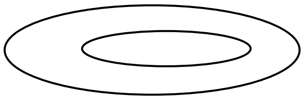
LEVEL 2 DFD:--


Search Query
New Medicine detail
Medicine List
Patient
View
Medicin
e
Medicine list
D2
Medicine
Ne
w
fee
d
list
Cname,cfeed,ex
perience
Doctor
Feedback
New Feedback detail
Search Query
Update user medicine
Search Query
Update
Medicine
View Patient
Response
Patient list
D3
Medicine
D1
Patientlogin

ENTITY-RELATIONSHIP MODEL

Entity-Relationship model is a type of database model based on the notion of real world entities and relationship among them. ER model creates a set of entities with their attributes, a set of constraints and relation among them. ER model is best used for the conceptual design of database.

Symbols of ER Diagram

NAME	SYMBOL	DESCRIPTION
Entity		An entity is a real word object.
Relationship		Relationship among two entities.
Attribute		Attribute are some of the properties of entity
Identifier Attribute		Attribute that define primary key;
Multivalued Attribute		All multivalued attribute are define

Mapping Cardinality- Mapping cardinalities define the number of associations between two entities. Mapping cardinalities:

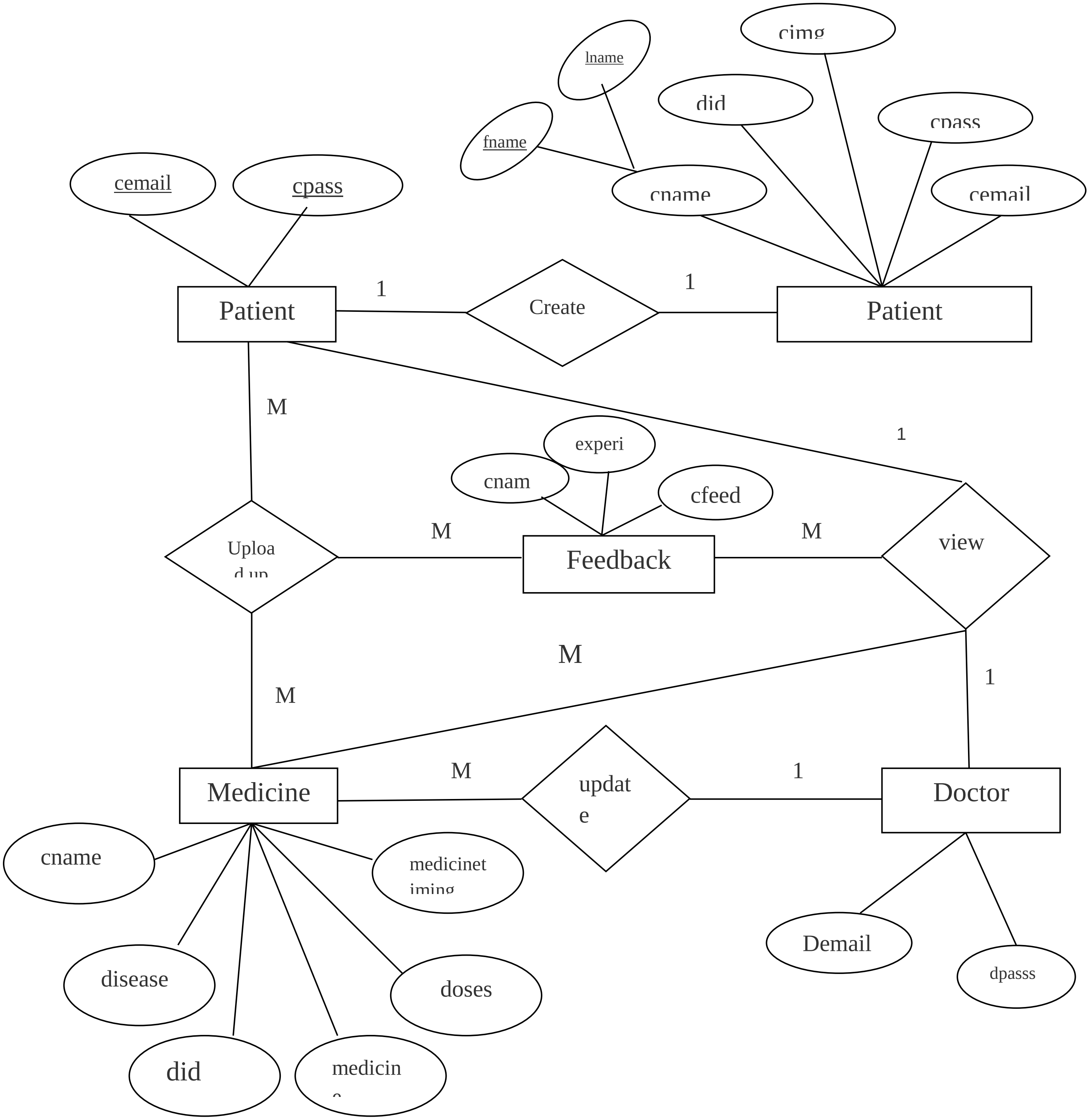
1. One-to-One

b>Many-to-One

c> One-to-Many

d> Many-to-Many

E-R Diagram for Medi-Info



SYSTEM DESIGN

The design of the system is the most critical factor affecting the quality of the software; it has major impact on the later phases, particularly testing and maintenance. The output of this phase is the design document. This document is similar to blueprint or plan for the solution.

The design activity is often divided into two phases: -

□ System Design

□ Detailed Design

System design aims to identify the modules that should be in the system, the specifications of these modules and how they interact with each other to produce the desired results. At the end of system design all the major data structures, file formats and the major modules in the system and their specifications are decided.

During Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase further details of the data structures and algorithmic design of each of the modules is specified. The logic of a module is usually specified in a high-level design description language, which is independent of the target language, in which the software will eventually be implemented.

MODULARIZATION

A software system is always divided into several sub-systems that make it easier for the development. A software system that is structured into several sub-systems makes it easy for the development and testing. The different sub-systems are known as the modules and the process of dividing an entire system into sub-systems is known as the modularization or decomposition.

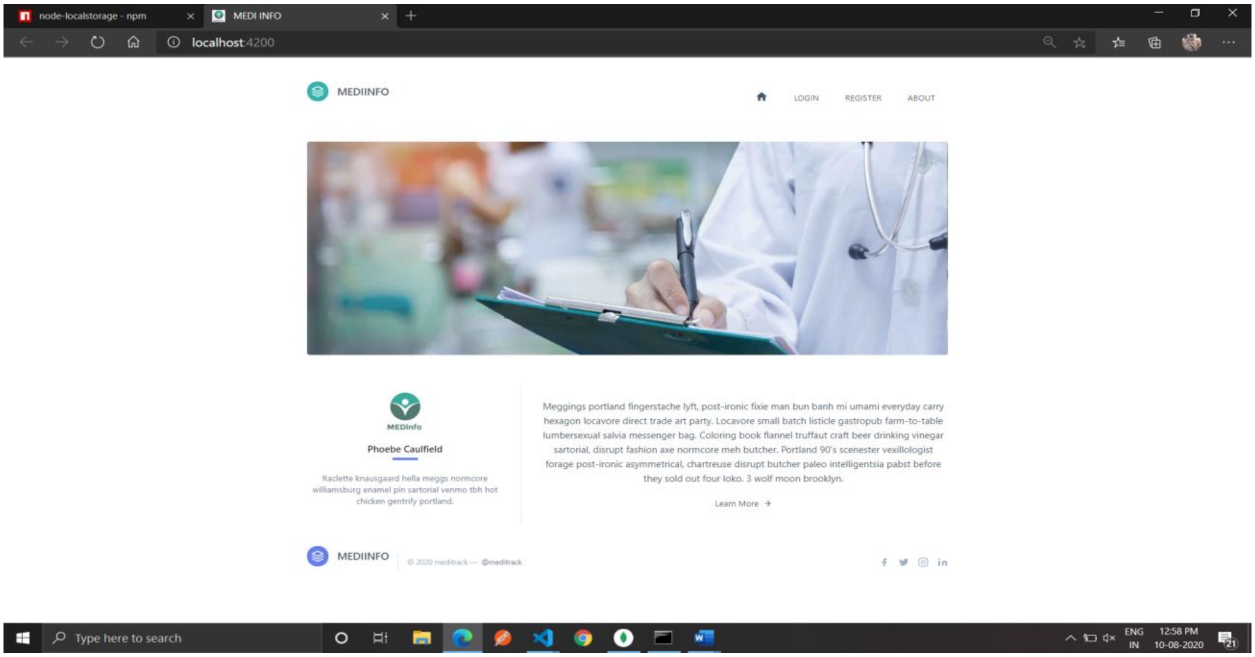
MODULE DESCRIPTION

Medi-Info Later is made up of Following modules. The modules are:

- VIEW MEDICINE
- ABOUT
- MY PROFILE
- HOME
- INSERT MEDICINE
- UPDATE MEDICINE
- FEEDBACK
- LOGOUT
- VIEW PATIENT
- VIEW FEEDBACK
- UPDATE MEDICINE RECORDS
- VIEW FEEDBACK

USER INTERFACE DESIGN

HOMEPAGE

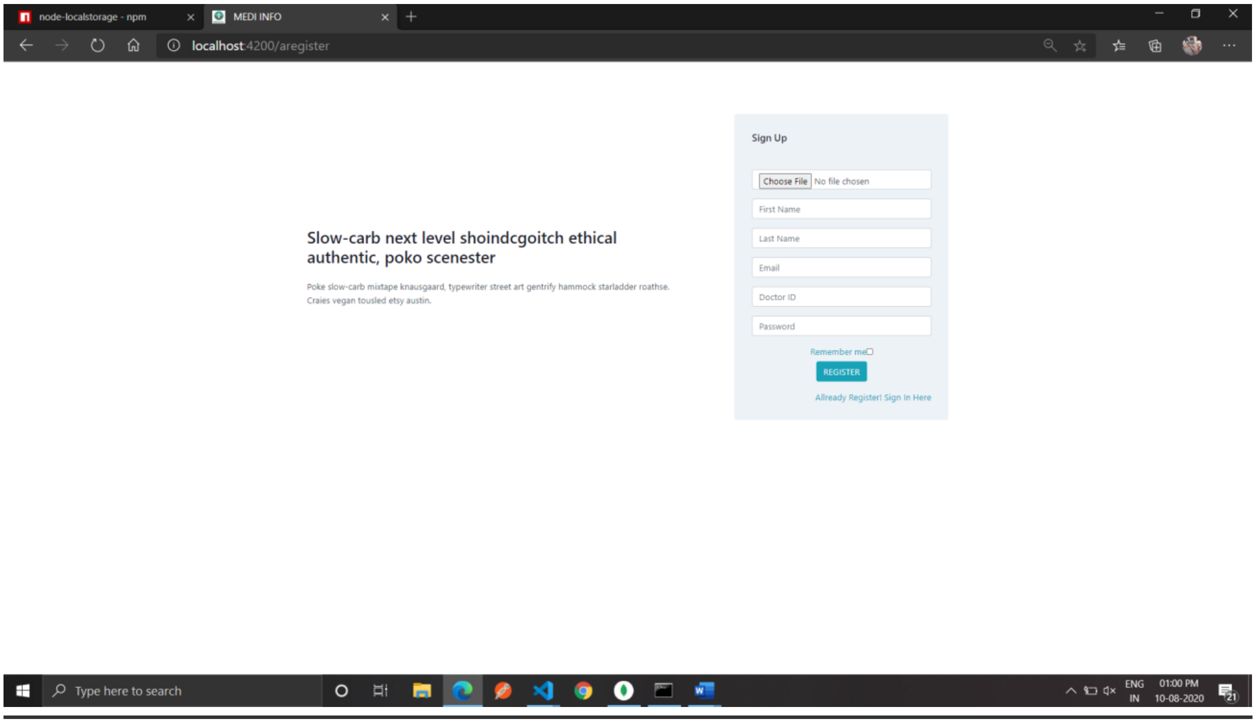


Phoebe Caulfield

Raclette knausgaard hella meggs normcore williamsburg enamel pin sartorial venmo tbh hot chicken gentrify portland.

Meggings portland fingerstache lyft, post-ironic fixie man bun banh mi umami everyday carry hexagon locavore direct trade art party. Locavore small batch listicle gastropub farm-to-table lumbersexual salvia messenger bag. Coloring book flannel truffaut craft beer drinking vinegar sartorial, disrupt fashion axe normcore meh butcher. Portland 90's scenester vexillologist forage post-ironic asymmetrical, chartreuse disrupt butcher paleo intelligentsia pabst before they sold out four loko. 3 wolf moon brooklyn.

Patient REGISTERATION:



Slow-carb next level shoindcgoitch ethical authentic, poko scenester

Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.

Sign Up

Choose File No file chosen

First Name

Last Name

Email

Doctor ID

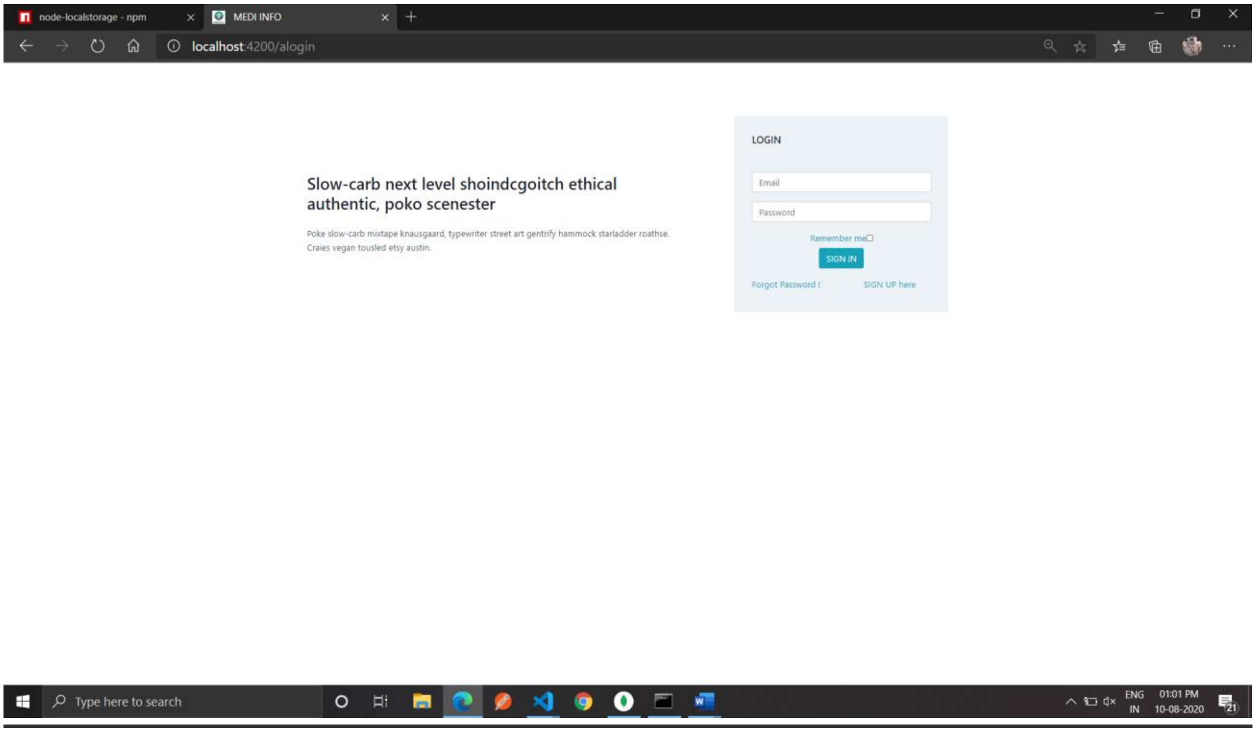
Password

Remember me□

REGISTER

Allready Register! Sign In Here

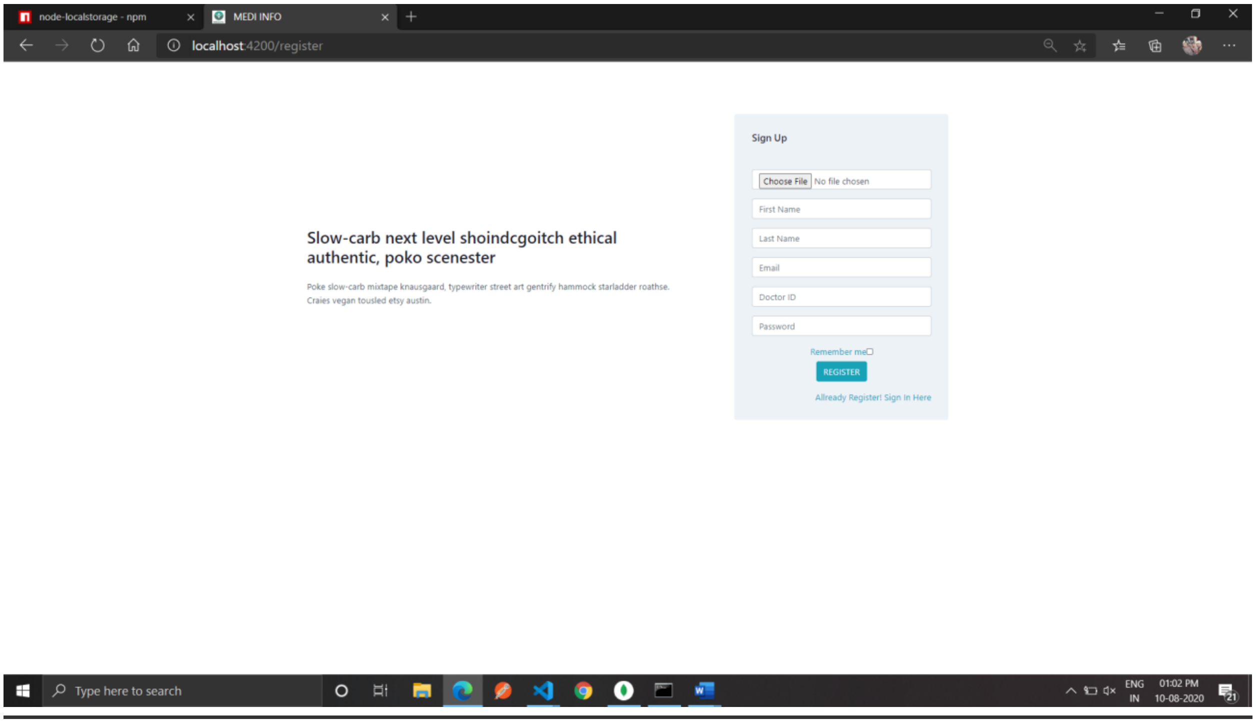
Patient LOGIN:



Slow-carb next level shoindcgoitch ethical authentic, poko scenester

Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.

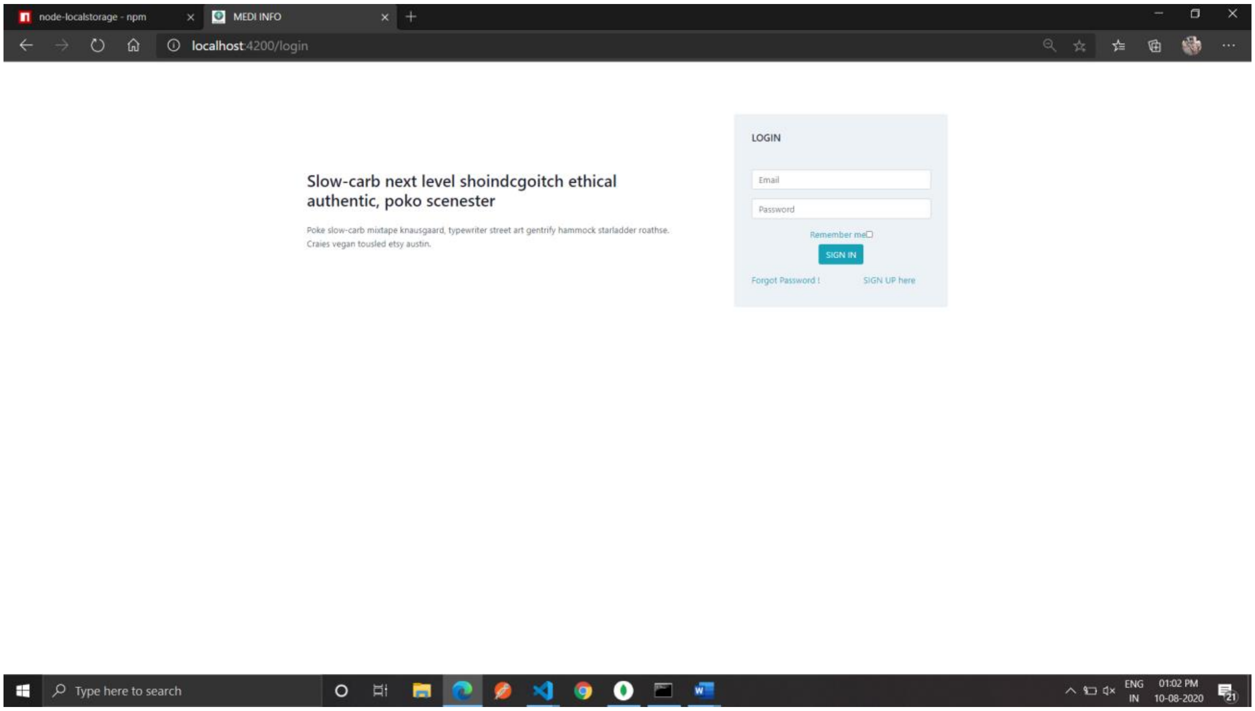
DOCTOR REGISTRATION:



Slow-carb next level shoindcgoitch ethical authentic, poko scenester

Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.

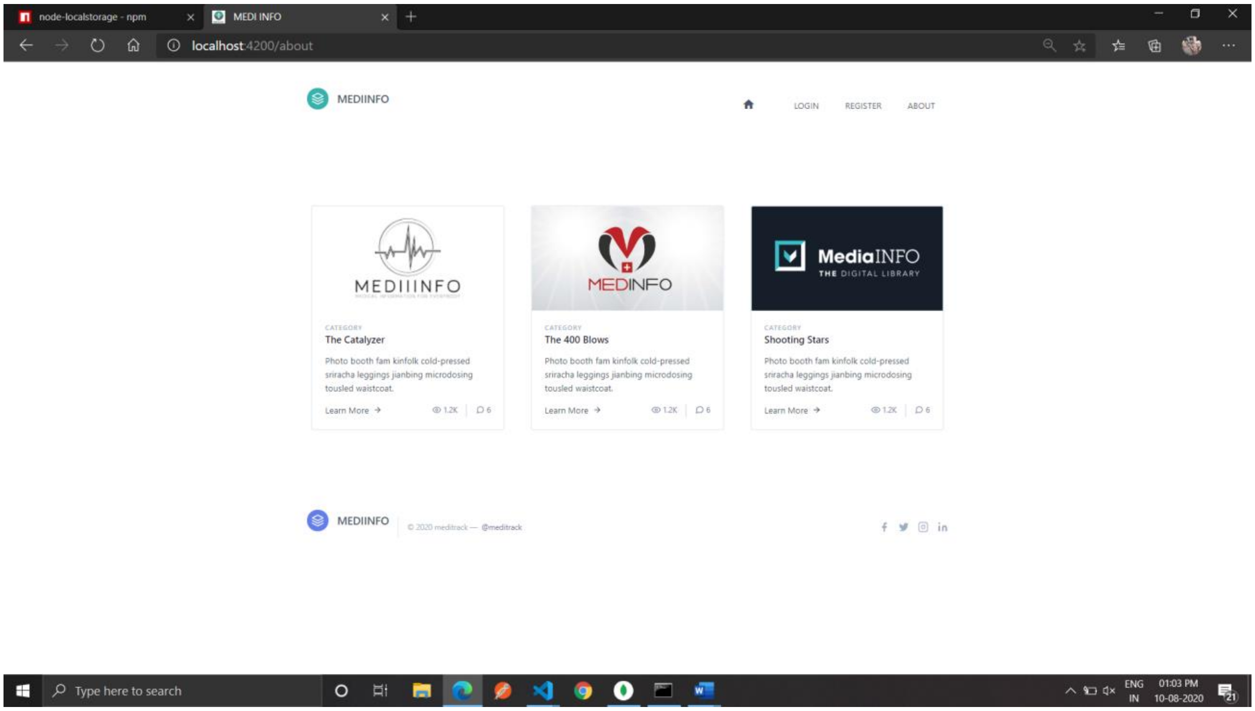
DOCTOR LOGIN:



Slow-carb next level shoindcgoitch ethical authentic, poko scenester

Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.

ABOUT



The Catalyzer

Photo booth fam kinfolk cold-pressed sriracha leggings jianbing microdosing tousled waistcoat.

The 400 Blows

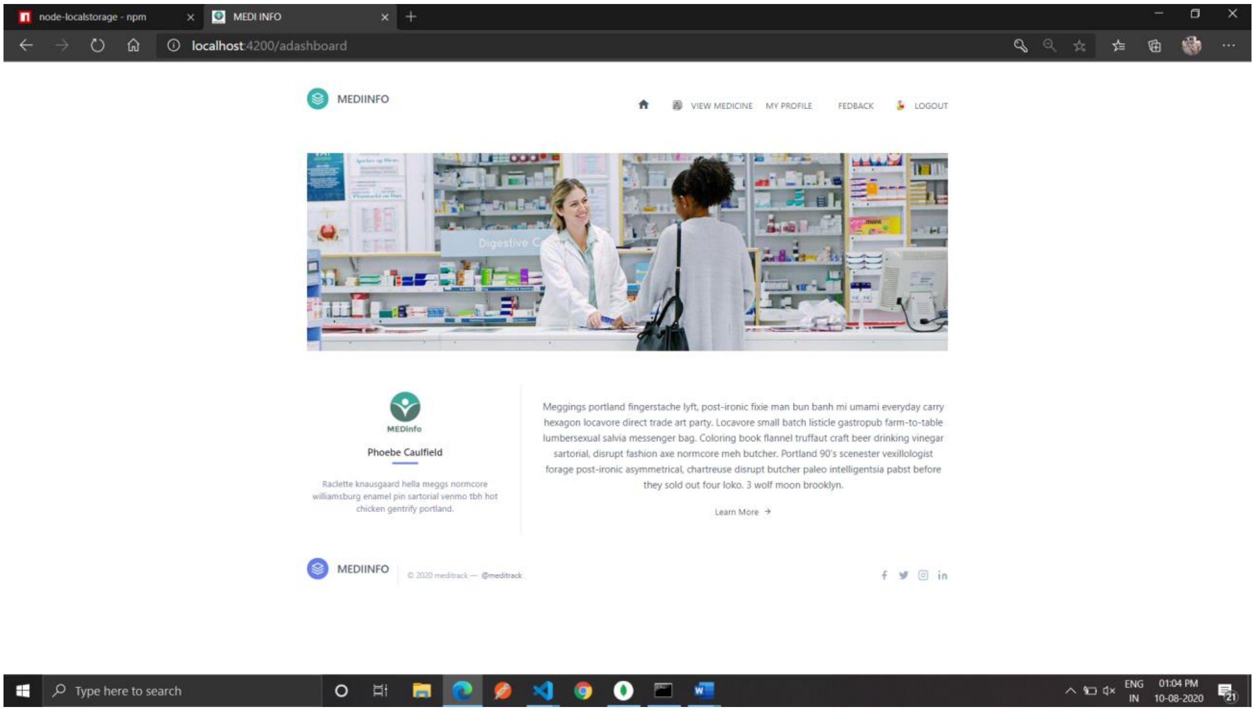
Photo booth fam kinfolk cold-pressed sriracha leggings jianbing microdosing tousled waistcoat.

Shooting Stars

Photo booth fam kinfolk cold-pressed sriracha leggings jianbing microdosing tousled waistcoat.

© 2020 meditrack — @meditrack

PATIENT DASHBOARD

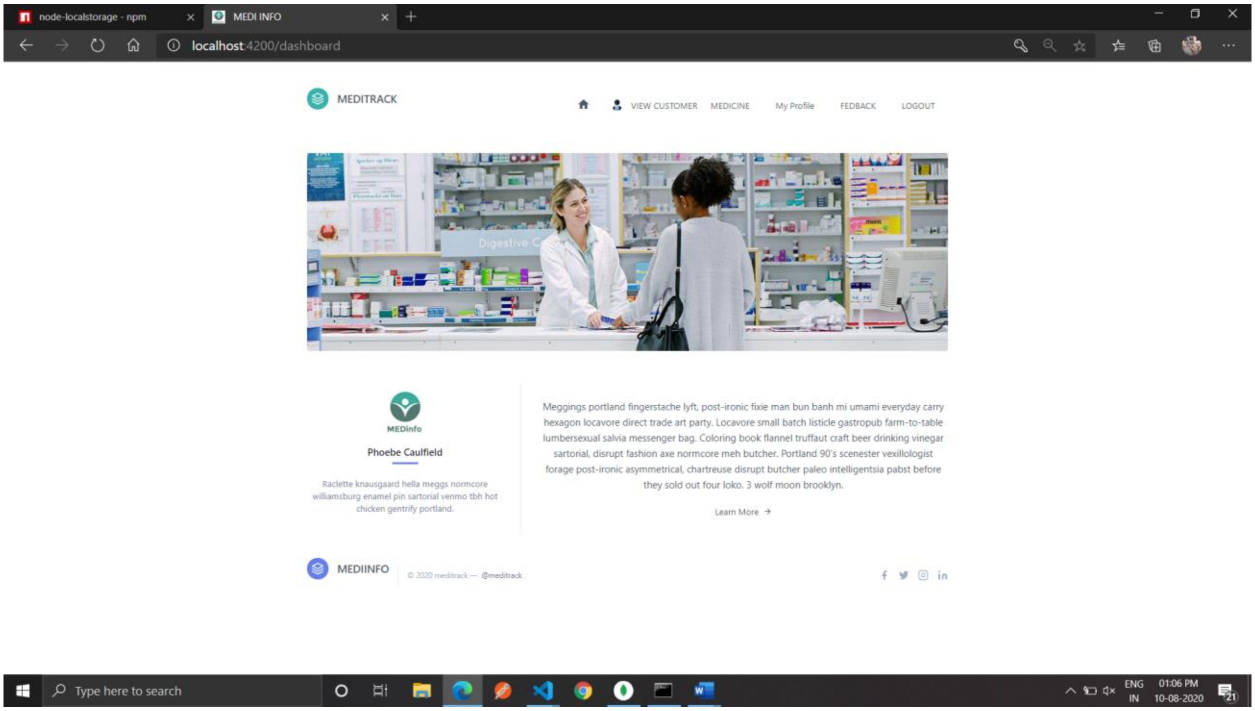


Phoebe Caulfield

Raclette knausgaard hella meggs normcore williamsburg enamel pin sartorial venmo tbh hot chicken gentrify portland.

Meggings portland fingerstache lyft, post-ironic fixie man bun banh mi umami everyday carry hexagon locavore direct trade art party. Locavore small batch listicle gastropub farm-to-table lumbersexual salvia messenger bag. Coloring book flannel truffaut craft beer drinking vinegar sartorial, disrupt fashion axe normcore meh butcher. Portland 90's scenester vexillologist forage post-ironic asymmetrical, chartreuse disrupt butcher paleo intelligentsia pabst before they sold out four loko. 3 wolf moon brooklyn.

DOCTOR DASHBOARD

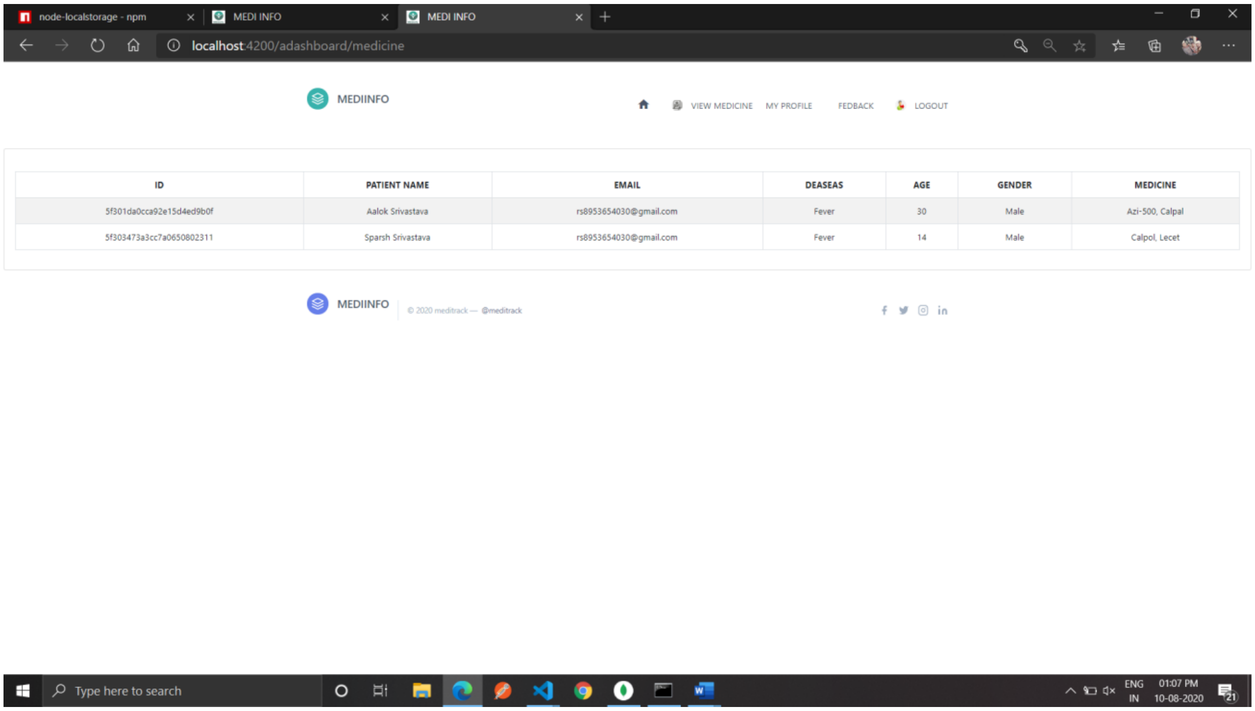


Phoebe Caulfield

Raclette knausgaard hella meggs normcore williamsburg enamel pin sartorial venmo tbh hot chicken gentrify portland.

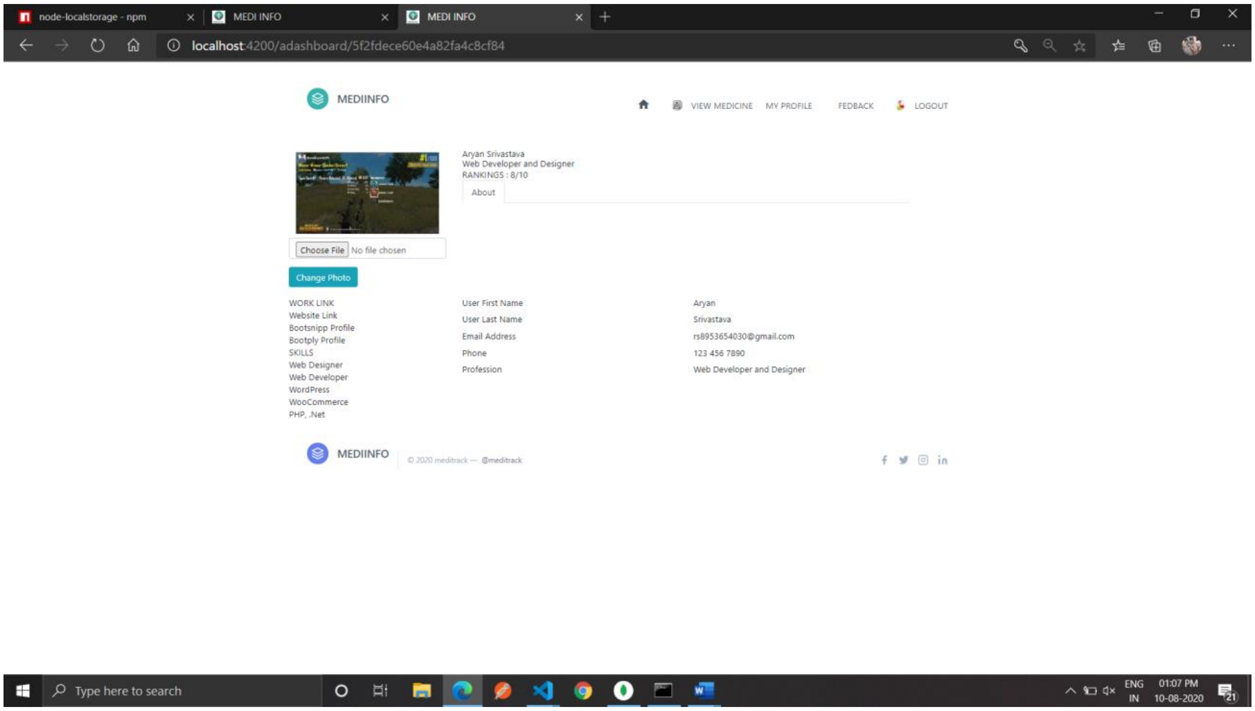
Meggings portland fingerstache lyft, post-ironic fixie man bun banh mi umami everyday carry hexagon locavore direct trade art party. Locavore small batch listicle gastropub farm-to-table lumbersexual salvia messenger bag. Coloring book flannel truffaut craft beer drinking vinegar sartorial, disrupt fashion axe normcore meh butcher. Portland 90's scenester vexillologist forage post-ironic asymmetrical, chartreuse disrupt butcher paleo intelligentsia pabst before they sold out four loko. 3 wolf moon brooklyn.

PATIENT VIEW MEDICINE



ID	PATIENT NAME	EMAIL	DEASEAS	AGE	GENDER	MEDICINE
5f301da0cca92e15d4ed9b0f	Aalok Srivastava	rs8953654030@gmail.com	Fever	30	Male	Azi-500, Calpal
5f303473a3cc7a0650802311	Sparsh Srivastava	rs8953654030@gmail.com	Fever	14	Male	Calpol, Lecet

MY PROFILE

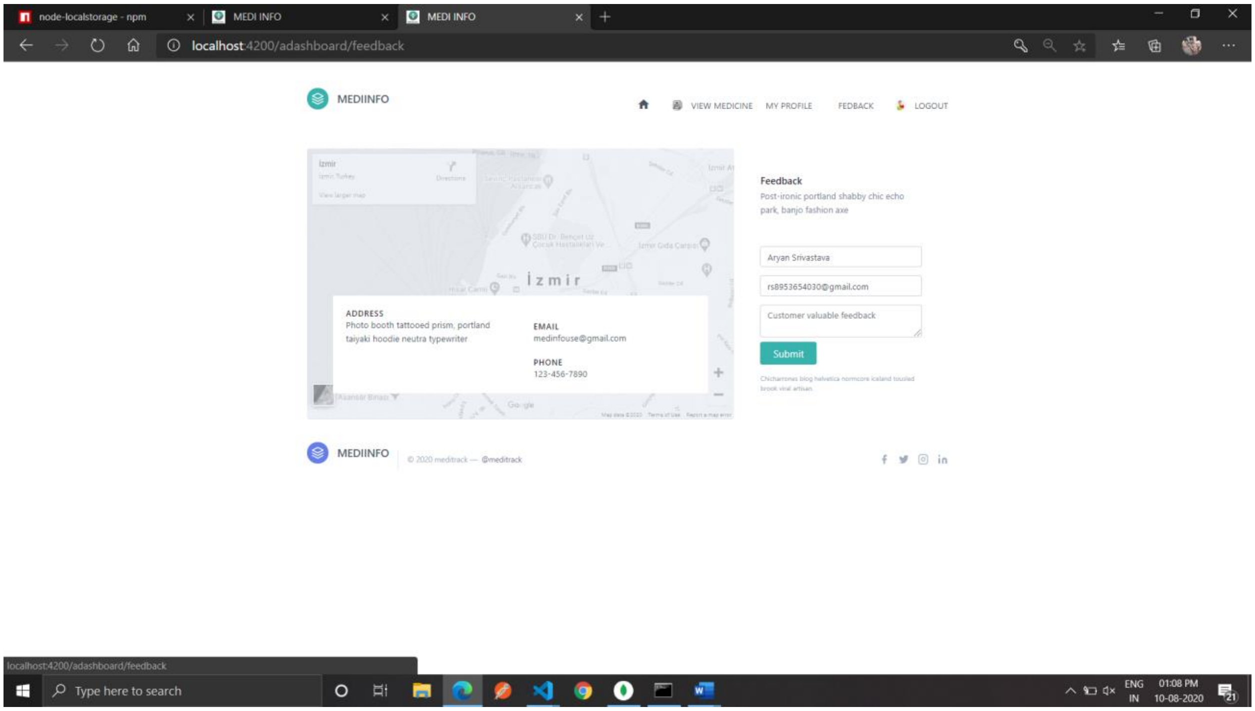


Aryan Srivastava
Web Developer and Designer
RANKINGS : 8/10

WORK LINK
Website Link
Bootsnipp Profile
Bootply Profile
SKILLS
Web Designer
Web Developer
WordPress
WooCommerce
PHP, .Net

User First Name	Aryan
User Last Name	Srivastava
Email Address	rs8953654030@gmail.com
Phone	123 456 7890
Profession	Web Developer and Designer

FEEDBACK



Feedback

Post-ironic portland shabby chic echo park, banjo fashion axe

Aryan Srivastava

rs8953654030@gmail.com

Customer valuable feedback

Chicharrones blog helvetica normcore iceland tousled brook viral artisan.

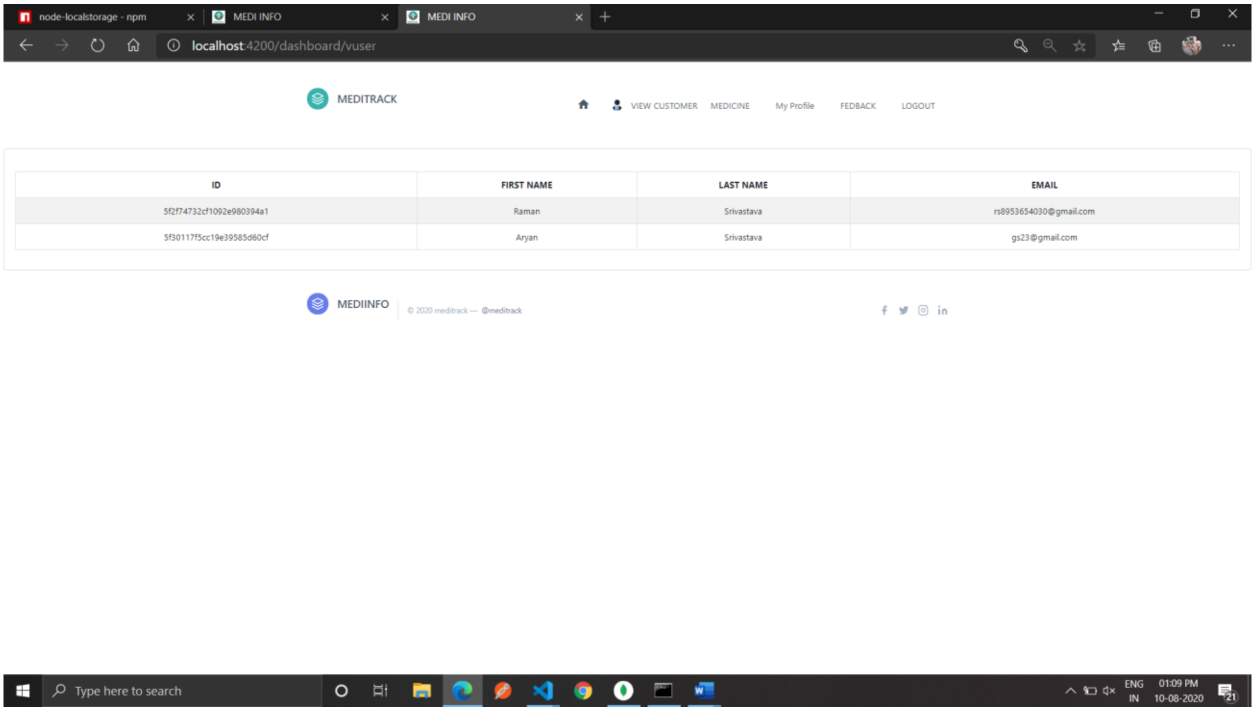
ADDRESS
Photo booth tattooed prism, portland taiyaki hoodie neutra typewriter

EMAIL
medinfouse@gmail.com

PHONE
123-456-7890

© 2020 meditrack — @meditrack

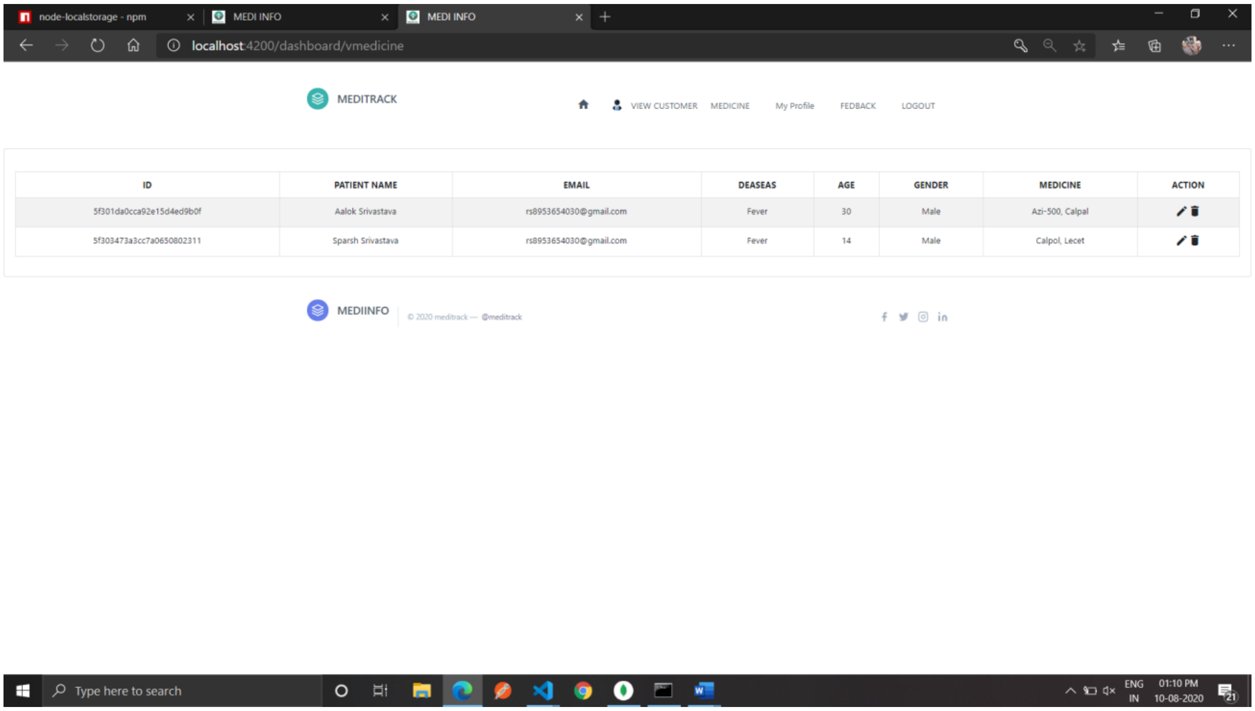
DOCTOR VIEW CUSTOMER



ID	FIRST NAME	LAST NAME	EMAIL
5f2f74732cf1092e980394a1	Raman	Srivastava	rs8953654030@gmail.com
5f30117f5cc19e39585d60cf	Aryan	Srivastava	gs23@gmail.com

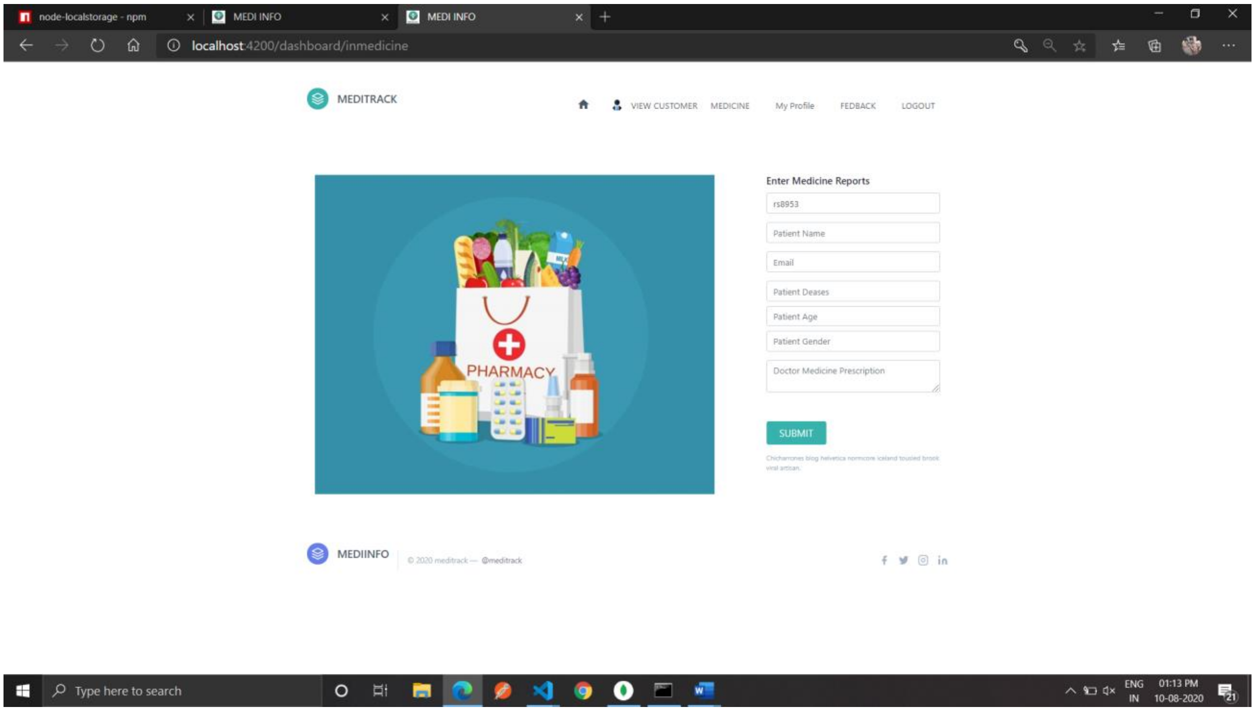
© 2020 meditrack — @meditrack

VIEW MEDICINE



ID	PATIENT NAME	EMAIL	DEASEAS	AGE	GENDER	MEDICINE	ACTION
5f301da0cca92e15d4ed9b0f	Aalok Srivastava	rs8953654030@gmail.com	Fever	30	Male	Azi-500, Calpal	✎ 🗑
5f303473a3cc7a0650802311	Sparsh Srivastava	rs8953654030@gmail.com	Fever	14	Male	Calpol, Lecet	✎ 🗑

INSERT MEDICINE



Enter Medicine Reports

rs8953

Patient Name

Email

Patient Deases

Patient Age

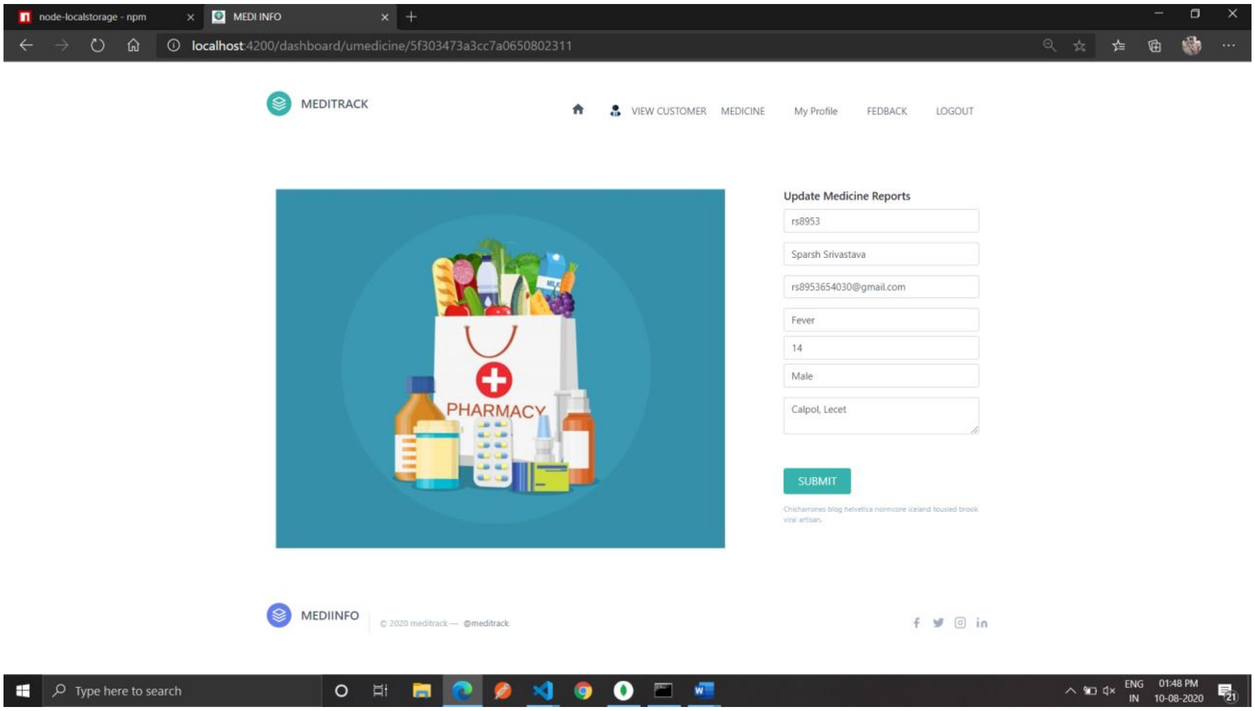
Patient Gender

Doctor Medicine Prescription

SUBMIT

Chicharrones blog helvetica normcore iceland tousled brook viral artisan.

UPDATE MEDICINE



Update Medicine Reports

rs8953

Sparsh Srivastava

rs8953654030@gmail.com

Fever

14

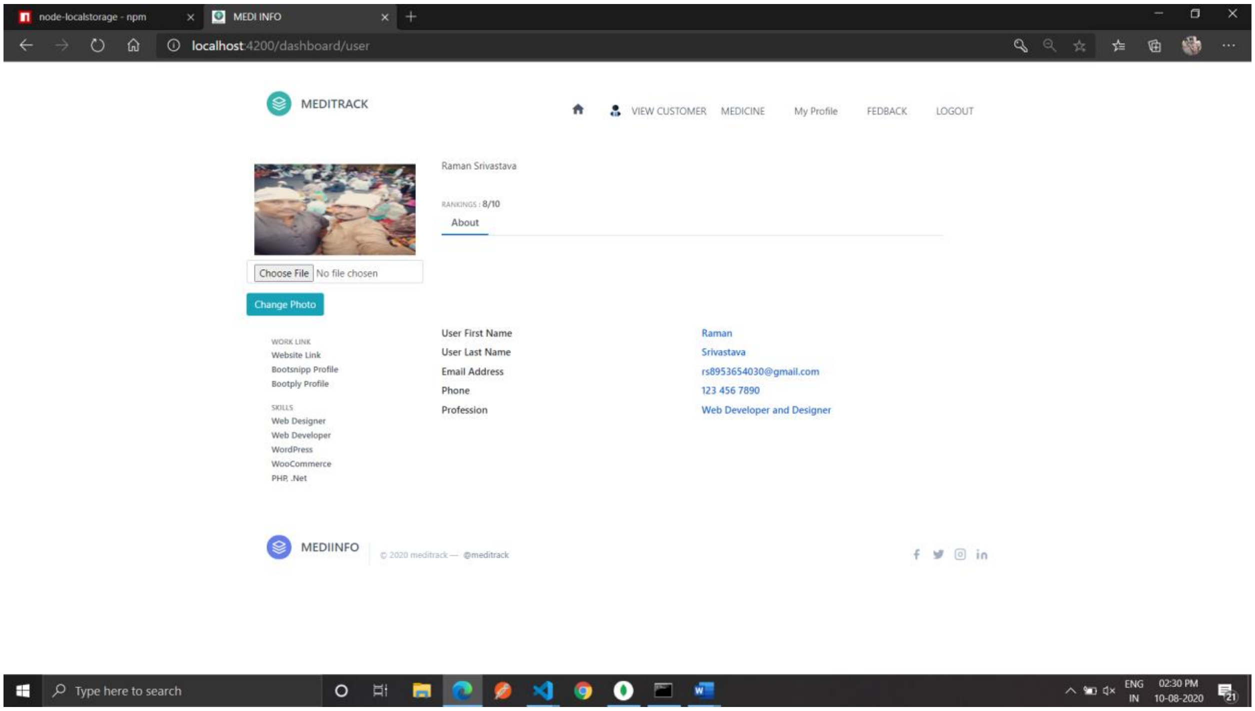
Male

Calpol, Lecet

SUBMIT

Chicharrones blog helvetica normcore iceland tousled brook viral artisan.

MY PROFILE



Raman Srivastava

RANKINGS : 8/10

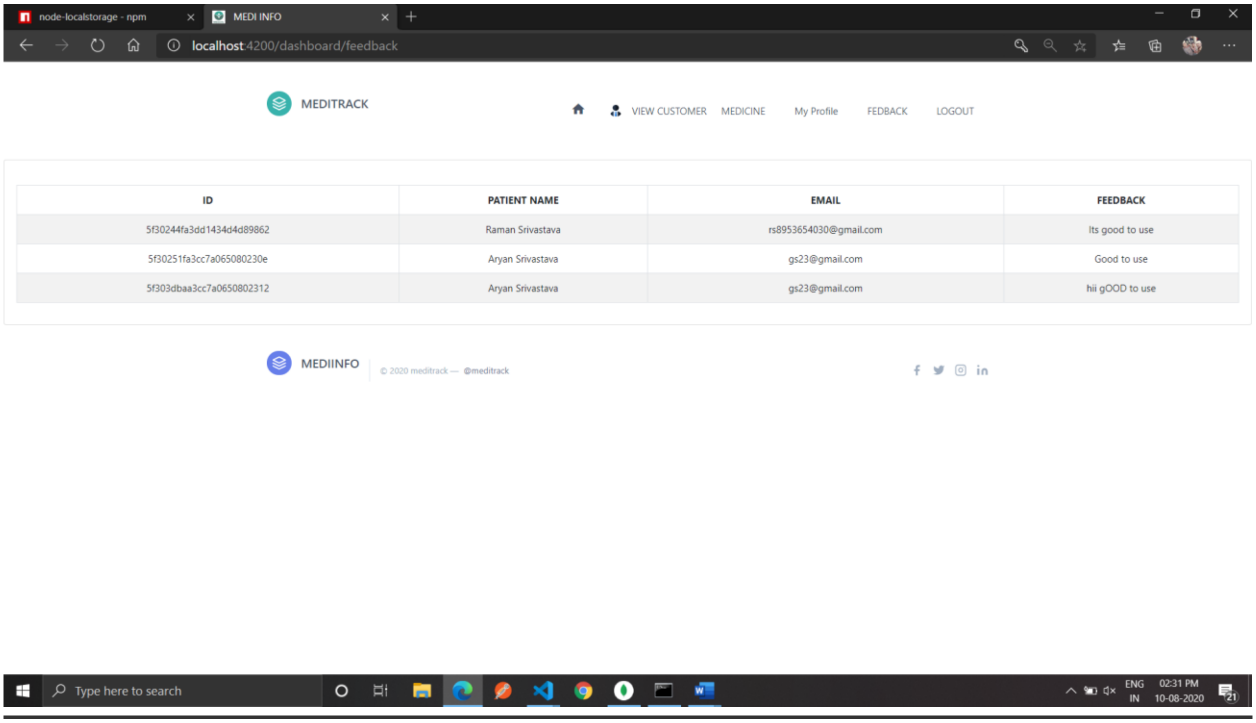
About

WORK LINK
Website Link
Bootsnipp Profile
Bootply Profile

SKILLS
Web Designer
Web Developer
WordPress
WooCommerce
PHP, .Net

User First Name	Raman
User Last Name	Srivastava
Email Address	rs8953654030@gmail.com
Phone	123 456 7890
Profession	Web Developer and Designer

FEEDBACK



ID	PATIENT NAME	EMAIL	FEEDBACK
5f30244fa3dd1434d4d89862	Raman Srivastava	rs8953654030@gmail.com	Its good to use
5f30251fa3cc7a065080230e	Aryan Srivastava	gs23@gmail.com	Good to use
5f303dbaa3cc7a0650802312	Aryan Srivastava	gs23@gmail.com	hii gOOD to use

CODING

➢ HTML PART :---

Main.html :--

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MEDI INFO</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="assets/logo.png">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">
  <link href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
      <!--Import materialize.css-->
      <link type="text/css" rel="stylesheet" href="css/materialize.min.css"  media="screen,projection"/>
      <!--Let browser know website is optimized for mobile-->
      <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
</head>
<body>
  <script type="text/javascript" src="js/materialize.min.js"></script>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI" crossorigin="anonymous"></script><app-root></app-root>
</body>
</html>
```

Home Component.html

```html
<app-header></app-header>
<section class="text-gray-700 body-font">
    <div class="container px-5 mt-2 mx-auto flex flex-col">
      <div class=" mx-auto">
```

```
        <div class="rounded-lg  overflow-hidden">
          <img alt="content" class="object-cover object-center h-full w-full" src="../../assets/7.png">
        </div>
        <div class="flex flex-col sm:flex-row mt-10">
          <div class="sm:w-1/3 text-center sm:pr-8 sm:py-8">
            <div class="w-20 h-20 rounded-full inline-flex items-center justify-center bg-gray-200 text-gray-400">
              <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" class="w-10 h-10" viewBox="0 0 24 24">
                <path d="M20 21v-2a4 4 0 00-4-4H8a4 4 0 00-4 4v2"></path>
                <circle cx="12" cy="7" r="4"></circle>
              </svg>
              <img src="../../assets/logo.png" alt="logo">
            </div>
            <div class="flex flex-col items-center text-center justify-center">
              <h2 class="font-medium title-font mt-4 text-gray-900 text-lg">Phoebe Caulfield</h2>
              <div class="w-12 h-1 bg-indigo-500 rounded mt-2 mb-4"></div>
              <p class="text-base text-gray-600">Raclette knausgaard hella meggs normcore williamsburg enamel pin sartorial venmo tbh hot chicken gentrify portland.</p>
            </div>
          </div>
          <div class="sm:w-2/3 sm:pl-8 sm:py-8 sm:border-l border-gray-300 sm:border-t-0 border-t mt-4 pt-4 sm:mt-0 text-center sm:text-left">
            <p class="leading-relaxed text-lg mb-4">Meggings portland fingerstache lyft, post-ironic fixie man bun banh mi umami everyday carry hexagon locavore direct trade art party. Locavore small batch listicle gastropub farm-to-table lumbersexual salvia messenger bag. Coloring book flannel truffaut craft beer drinking vinegar sartorial, disrupt fashion axe normcore meh butcher. Portland 90's scenester vexillologist forage post-ironic asymmetrical, chartreuse disrupt butcher paleo intelligentsia pabst before they sold out four loko. 3 wolf moon brooklyn.</p>
            <a class="text-indigo-500 inline-flex items-center">Learn More
              <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" class="w-4 h-4 ml-2" viewBox="0 0 24 24">
                <path d="M5 12h14M12 5l7 7-7 7"></path>
              </svg>
            </a>
          </div>
        </div>
      </div>
    </div>
  </section>
<app-footer></app-footer>
```

Aregister.compoment.html

```
<div  *ngIf="!param" id="myDIV" class="wrapper">
 <section class="text-gray-700 body-font">
  <div class="container px-5 py-24 mx-auto flex flex-wrap items-center">
   <div class="lg:w-3/5 md:w-1/2 md:pr-16 lg:pr-0 pr-0">
    <h1 class="title-font font-medium text-3xl text-gray-900">Slow-carb next level shoindcgoitch ethical authentic, poko scenester</h1>
    <p class="leading-relaxed mt-4">Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.</p>
   </div>
   <div class="lg:w-2/6 md:w-1/2 bg-gray-200 rounded-lg p-8 flex flex-col md:ml-auto w-full mt-10 md:mt-0">
    <form [formGroup]="profileForm" (ngSubmit)="onSubmit()" class="form" enctype="multipart/form-data" >
    <h2 class="text-gray-900 text-lg font-medium title-font mb-5">Sign Up</h2>
    <div class="form-group">
     <input type="file" class="form-control" formControlName="aimg" (change)="onFileSelect($event)" />
    </div>
    <div class="form-group">
     <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4"  placeholder="First Name" type="text" class="form-control" formControlName="f_name" required minlength="3">
     <small *ngIf="f_name.invalid && (profileForm.controls['f_name'].dirty || profileForm.controls['f_name'].touched)" class="text-danger">
      <small *ngIf="f_name.errors.required" class="text-danger mx-auto">First Name required</small>
      <small *ngIf="f_name.errors.minlength" class="text-danger">First Name should contain minimum three character</small>
     </small>
    </div>
    <div class="form-group">
     <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Last Name" type="text" class="form-control" formControlName="l_name"  required minlength="2">
     <small *ngIf="l_name.invalid && (profileForm.controls['l_name'].dirty || profileForm.controls['l_name'].touched)" class="text-danger">
      <small *ngIf="l_name.errors.required" class="text-danger">Last Name required</small>
      <small *ngIf="l_name.errors.minlength" class="text-danger">Last Name should contain minimum two character</small>
     </small>
    </div>
    <div class="form-group">
     <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Email" type="email" class="form-control" formControlName="email" required email>
     <small *ngIf="email.invalid && (profileForm.controls['email'].dirty || profileForm.controls['email'].touched)" class="text-danger">
      <small *ngIf="email.errors.required" class="text-danger">Email required</small>
      <small *ngIf="email.errors.email" class="text-danger">Email is be Valid</small>
     </small>
    </div>
    <div class="form-group">
     <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Doctor ID" type="text" class="form-control" formControlName="did" required minlength="3">
     <small *ngIf="did.invalid && (profileForm.controls['did'].dirty || profileForm.controls['did'].touched)" class="text-danger">
      <small *ngIf="did.errors.required" class="text-danger">Doctor Id required</small>
      <small *ngIf="did.errors.minlength" class="text-danger">Doctor Id is be Valid</small>
     </small>
    </div>
    <div class="form-group">
     <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Password" type="password" class="form-control" formControlName="password" required minlength="8">
     <small *ngIf="password.invalid && (profileForm.controls['password'].dirty || profileForm.controls['password'].touched)" class="text-danger">
      <small *ngIf="password.errors.required" class="text-danger">Password required</small>
      <small *ngIf="password.errors.minlength" class="text-danger">Password must be strong</small>
     </small>
    </div>
    <div class="form-group">
     <div class="form-group" style="text-align: center;">
      <label for="remember-me" class="text-info"><span>Remember me</span> <span><input id="remember-me" name="remember-me" type="checkbox"></span></label><br>
      <input type="submit" class="btn btn-info btn-md" value="REGISTER">
     </div>
    </div>
   </form>
   <div id="register-link" class="text-right">
    <a routerLink='/alogin' class="text-info">Allready Register! Sign In Here</a>
   </div>
  </div>
 </div>
 </section>
</div>
<div  *ngIf="param" id="myDIV" class="wrapper" >
 <section class="text-gray-700 body-font">
  <div class="container px-5 py-24 mx-auto flex flex-wrap items-center">
```

```
      <div class="lg:w-3/5 md:w-1/2 md:pr-16 lg:pr-0 pr-0">
        <h1 class="title-font font-medium text-3xl text-gray-900">Slow-carb next level shoindcgoitch ethical authentic, poko scenester</h1>
        <p class="leading-relaxed mt-4">Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.</p>
      </div>
      <div class="lg:w-2/6 md:w-1/2 bg-gray-200 rounded-lg p-8 flex flex-col md:ml-auto w-full mt-10 md:mt-0">
        <form [formGroup]="otpForm" (ngSubmit)="onSubmitotp()" class="form" >
        <h2 class="text-gray-900 text-lg font-medium title-font mb-5">Otp Verified</h2>
        <div class="form-group">
          <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Email" type="email" class="form-control" formControlName="emails" [(ngModel)]="emai" required email>
          <small *ngIf="emails.invalid && (otpForm.controls['emails'].dirty || otpForm.controls['emails'].touched)" class="text-danger">
            <small *ngIf="emails.errors.required" class="text-danger">Email required</small>
            <small *ngIf="emails.errors.email" class="text-danger">Email is be Valid</small>
          </small>
        </div>
      <div class="form-group">
        <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="OTP" type="text" class="form-control" formControlName="otp" required minlength="6">
        <small *ngIf="otp.invalid && (otpForm.controls['otp'].dirty || otpForm.controls['otp'].touched)" class="text-danger">
          <small *ngIf="otp.errors.required" class="text-danger"> OTP required</small>
        </small>
    </div>
      <div class="form-group">
        <div class="form-group" style="text-align: center;">
          <input type="submit" class="btn btn-info btn-md" value="submit">
      </div>
      </div>
    </form>
      </div>
    </div>
  </section>
</div>
```

PRegister.component.html : --

```
<div  *ngIf="!param" id="myDIV" class="wrapper" >
  <section class="text-gray-700 body-font">
    <div class="container px-5 py-24 mx-auto flex flex-wrap items-center">
      <div class="lg:w-3/5 md:w-1/2 md:pr-16 lg:pr-0 pr-0">
        <h1 class="title-font font-medium text-3xl text-gray-900">Slow-carb next level shoindcgoitch ethical authentic, poko scenester</h1>
        <p class="leading-relaxed mt-4">Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.</p>
      </div>
      <div class="lg:w-2/6 md:w-1/2 bg-gray-200 rounded-lg p-8 flex flex-col md:ml-auto w-full mt-10 md:mt-0">
        <form [formGroup]="profileForm" (ngSubmit)="onSubmit()" class="form" enctype="multipart/form-data">
        <h2 class="text-gray-900 text-lg font-medium title-font mb-5">Sign Up</h2>
        <div class="form-group">
          <input type="file" class="form-control" formControlName="uimg" (change)="onFileSelect($event)" />
        </div>
        <div class="form-group">
          <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4"  placeholder="First Name" type="text" class="form-control" formControlName="f_name" required minlength="3">
          <small *ngIf="f_name.invalid && (profileForm.controls['f_name'].dirty || profileForm.controls['f_name'].touched)" class="text-danger">
            <small *ngIf="f_name.errors.required" class="text-danger mx-auto">First Name required</small>
            <small *ngIf="f_name.errors.minlength" class="text-danger">First Name should contain minimum three character</small>
          </small>
        </div>
        <div class="form-group">
          <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Last Name" type="text" class="form-control" formControlName="l_name"  required minlength="2">
          <small *ngIf="l_name.invalid && (profileForm.controls['l_name'].dirty || profileForm.controls['l_name'].touched)" class="text-danger">
            <small *ngIf="l_name.errors.required" class="text-danger">Last Name required</small>
            <small *ngIf="l_name.errors.minlength" class="text-danger">Last Name should contain minimum two character</small>
          </small>
        </div>
        <div class="form-group">
          <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Email" type="email" class="form-control" formControlName="email" required email>
```

```
     <small *ngIf="email.invalid && (profileForm.controls['email'].dirty || profileForm.controls['email'].touched)" class="text-danger">
      <small *ngIf="email.errors.required" class="text-danger">Email required</small>
      <small *ngIf="email.errors.email" class="text-danger">Email is be Valid</small>
     </small>
    </div>
    <div class="form-group">
     <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Doctor ID" type="text" class="form-control" formControlName="did" required minlength="3">
     <small *ngIf="did.invalid && (profileForm.controls['did'].dirty || profileForm.controls['did'].touched)" class="text-danger">
      <small *ngIf="did.errors.required" class="text-danger">Doctor Id required</small>
      <small *ngIf="did.errors.minlength" class="text-danger">Doctor Id is be Valid</small>
     </small>
    </div>
    <div class="form-group">
     <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Password" type="password" class="form-control" formControlName="password" required minlength="8">
     <small *ngIf="password.invalid && (profileForm.controls['password'].dirty || profileForm.controls['password'].touched)" class="text-danger">
      <small *ngIf="password.errors.required" class="text-danger">Password required</small>
      <small *ngIf="password.errors.minlength" class="text-danger">Password must be strong</small>
     </small>
   </div>
   <div class="form-group">
    <div class="form-group" style="text-align: center;">
     <label for="remember-me" class="text-info"><span>Remember me</span> <span><input id="remember-me" name="remember-me" type="checkbox"></span></label><br>
     <input type="submit" class="btn btn-info btn-md" value="REGISTER">
   </div>
   </div>
  </form>
    <div id="register-link" class="text-right">
     <a routerLink='/login' class="text-info">Allready Register! Sign In Here</a>
    </div>
   </div>
  </div>
 </section>
</div>
<div  *ngIf="param" id="myDIV" class="wrapper" >
 <section class="text-gray-700 body-font">
  <div class="container px-5 py-24 mx-auto flex flex-wrap items-center">
```

```
      <div class="lg:w-3/5 md:w-1/2 md:pr-16 lg:pr-0 pr-0">
        <h1 class="title-font font-medium text-3xl text-gray-900">Slow-carb next level shoindcgoitch ethical authentic, poko scenester</h1>
        <p class="leading-relaxed mt-4">Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.</p>
      </div>
      <div class="lg:w-2/6 md:w-1/2 bg-gray-200 rounded-lg p-8 flex flex-col md:ml-auto w-full mt-10 md:mt-0">
        <form [formGroup]="otpForm" (ngSubmit)="onSubmitotp()" class="form" >
        <h2 class="text-gray-900 text-lg font-medium title-font mb-5">Otp Verified</h2>
        <div class="form-group">
          <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Email" type="email" class="form-control" formControlName="emails" [(ngModel)]="emai" required email>
          <small *ngIf="emails.invalid && (otpForm.controls['emails'].dirty || otpForm.controls['emails'].touched)" class="text-danger">
            <small *ngIf="emails.errors.required" class="text-danger">Email required</small>
            <small *ngIf="emails.errors.email" class="text-danger">Email is be Valid</small>
          </small>
        </div>
      <div class="form-group">
        <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="OTP" type="text" class="form-control" formControlName="otp" required minlength="6">
        <small *ngIf="otp.invalid && (otpForm.controls['otp'].dirty || otpForm.controls['otp'].touched)" class="text-danger">
          <small *ngIf="otp.errors.required" class="text-danger"> OTP required</small>
        </small>
    </div>
      <div class="form-group">
        <div class="form-group" style="text-align: center;">
          <input type="submit" class="btn btn-info btn-md" value="submit">
      </div>
      </div>
    </form>
      </div>
    </div>
  </section>
</div>
```

Alogin.component.js

```
<section class="text-gray-700 body-font">
  <div class="container px-5 py-24 mx-auto flex flex-wrap items-center">
   <div class="lg:w-3/5 md:w-1/2 md:pr-16 lg:pr-0 pr-0">
    <h1 class="title-font font-medium text-3xl text-gray-900">Slow-carb next level shoindcgoitch ethical authentic, poko scenester</h1>
    <p class="leading-relaxed mt-4">Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.</p>
   </div>
   <div class="lg:w-2/6 md:w-1/2 bg-gray-200 rounded-lg p-8 flex flex-col md:ml-auto w-full mt-10 md:mt-0">
    <form [formGroup]="profileForm" (ngSubmit)="onSubmit()" class="form" >
     <h2 class="text-gray-900 text-lg font-medium title-font mb-5">LOGIN</h2>
     <div class="form-group">
      <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Email" type="email" class="form-control" formControlName="email" required email>
      <small *ngIf="email.invalid && (profileForm.controls['email'].dirty || profileForm.controls['email'].touched)" class="text-danger">
       <small *ngIf="email.errors.required" class="text-danger">Email required</small>
       <small *ngIf="email.errors.email" class="text-danger">Email is be Valid</small>
      </small>
     </div>
     <div class="form-group">
      <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Password" type="password" class="form-control" formControlName="password" required minlength="8">
      <small *ngIf="password.invalid && (profileForm.controls['password'].dirty || profileForm.controls['password'].touched)" class="text-danger">
       <small *ngIf="password.errors.required" class="text-danger">Password required</small>
       <small *ngIf="password.errors.minlength" class="text-danger">Password must be strong</small>
      </small>
     </div>
     <div class="form-group">
      <div class="form-group" style="text-align: center;">
      <label for="remember-me" class="text-info"><span>Remember me</span> <span><input id="remember-me" name="remember-me" type="checkbox"></span></label><br>
      <input type="submit" class="btn btn-info btn-md" value="SIGN IN">
     </div>
    </div>
   <label class="text-info" style="float: left;">
    <span style="float: left;" onclick="frgpass()" data-toggle="modal" data-target="#myModal">Forgot Password !</span>
```

```
        <div class="modal fade" id="myModal" role="dialog">
          <div class="modal-dialog">
              <!-- Modal content-->
            <div class="modal-content">
              <div class="modal-body">
                <div class="form-group">
                  <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Email" type="email" class="form-control" formControlName="email" required email>
                  <small *ngIf="email.invalid && (email.dirty || email.touched)" class="text-danger">
                    <small *ngIf="email.errors.required" class="text-danger">Email required</small>
                    <small *ngIf="email.errors.email" class="text-danger">Email is be Valid</small>
                  </small>
                </div>
              </div>
              <div class="modal-footer">
                <button type="button" class="btn btn-default" (click)="frgpass()">Save</button>
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>                </div>
            </div>
          </div>
        </div>
        <div id="register-link" style="float: right; margin-left: 80px;">
          <a routerLink='/aregister' class="text-info">SIGN UP here</a>
        </div>
      </label>
        </form>
      </div>
  </div>
</section>
```

PLogin.component.html :--

```
<section class="text-gray-700 body-font">
    <div class="container px-5 py-24 mx-auto flex flex-wrap items-center">
      <div class="lg:w-3/5 md:w-1/2 md:pr-16 lg:pr-0 pr-0">
        <h1 class="title-font font-medium text-3xl text-gray-900">Slow-carb next level shoindcgoitch ethical authentic, poko scenester</h1>
        <p class="leading-relaxed mt-4">Poke slow-carb mixtape knausgaard, typewriter street art gentrify hammock starladder roathse. Craies vegan tousled etsy austin.</p>
      </div>
      <div class="lg:w-2/6 md:w-1/2 bg-gray-200 rounded-lg p-8 flex flex-col md:ml-auto w-full mt-10 md:mt-0">
        <form [formGroup]="profileForm" (ngSubmit)="onSubmit()" class="form" >
        <h2 class="text-gray-900 text-lg font-medium title-font mb-5">LOGIN</h2>
        <div class="form-group">
         <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Email" type="email" class="form-control" formControlName="email" required email>
         <small *ngIf="email.invalid && (profileForm.controls['email'].dirty || profileForm.controls['email'].touched)" class="text-danger">
          <small *ngIf="email.errors.required" class="text-danger">Email required</small>
          <small *ngIf="email.errors.email" class="text-danger">Email is be Valid</small>
         </small>
        </div>
        <div class="form-group">
         <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Password" type="password" class="form-control" formControlName="password" required minlength="8">
         <small *ngIf="password.invalid && (profileForm.controls['password'].dirty || profileForm.controls['password'].touched)" class="text-danger">
          <small *ngIf="password.errors.required" class="text-danger">Password required</small>
          <small *ngIf="password.errors.minlength" class="text-danger">Password must be strong</small>
         </small>
      </div>
      <div class="form-group">
        <div class="form-group" style="text-align: center;">
         <label for="remember-me" class="text-info"><span>Remember me</span> <span><input id="remember-me" name="remember-me" type="checkbox"></span></label><br>
         <input type="submit" class="btn btn-info btn-md" value="SIGN IN">
      </div>
      </div>
      <label class="text-info" style="float: left;">
        <span style="float: left;" onclick="frgpass()" data-toggle="modal" data-target="#myModal">Forgot Password !</span>

        <div class="modal fade" id="myModal" role="dialog">
         <div class="modal-dialog">
           <!-- Modal content-->
          <div class="modal-content">
           <div class="modal-body">
            <div class="form-group">
             <input class="bg-white rounded border border-gray-400 focus:outline-none focus:border-indigo-500 text-base px-4 py-2 mb-4" placeholder="Email" type="email" class="form-control" formControlName="email" required email>
             <small *ngIf="email.invalid && (email.dirty || email.touched)" class="text-danger">
              <small *ngIf="email.errors.required" class="text-danger">Email required</small>
              <small *ngIf="email.errors.email" class="text-danger">Email is be Valid</small>
             </small>
            </div>
           </div>
           <div class="modal-footer">
            <button type="button" class="btn btn-default" (click)="frgpass()">Submit</button>
            <button type="button" class="btn btn-default" data-
    dismiss="modal">Close</button>               </div>
          </div>
         </div>
        </div>
        <div id="register-link" style="float: right; margin-left: 80px;">
         <a routerLink='/register' class="text-info">SIGN UP here</a>
        </div>
      </label>
     </form>
      </div>
     </div>
    </section>
```

➢ Component.js file :--

Alogin :--

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormBuilder } from '@angular/forms';
import {Validators} from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from '../../auth.service';
import { ToastrService } from 'ngx-toastr';
import { CookieService } from 'ngx-cookie-service';

@Component({
  selector: 'app-alogin',
  templateUrl: './alogin.component.html',
  styleUrls: ['./alogin.component.css']
})
export class AloginComponent implements OnInit {

  profileForm = this.fb.group({
    email : ['', Validators.required],
    password : ['', Validators.required],
  });
  // tslint:disable-next-line: max-line-length
  constructor(private fb: FormBuilder, private router: Router, private service: AuthService, private toast: ToastrService, private cookie: CookieService) { }

  get email() {return this.profileForm.get('email'); }
  get password() {return this.profileForm.get('password'); }

  addCookie(data: any) {
    this.cookie.set('Authorization', data );
  }
  getCookie() {
    return this.cookie.get('Authorization');
  }

  onSubmit() {
    if (this.profileForm.invalid) {
      this.profileForm.get('email').markAsTouched();
      this.profileForm.get('password').markAsTouched();
    // tslint:disable-next-line: one-line
    }else{
      this.service.loginAdmin({
        email: this.profileForm.value.email,
        password: this.profileForm.value.password
      }).subscribe((res) => {
        this.addCookie(res.token);
        this.cookie.set('id', res.data);
        this.showtoasterdf(res);
      });
    }
  }
  showtoasterdf(res) {
    // tslint:disable-next-line: whitespace
    if(res.sucess) {
      this.toast.success(res.sucess, res.message);
      this.router.navigate([`${'/adashboard'}`]);
    // tslint:disable-next-line: one-line
    }else{
      this.toast.warning(res.sucess, res.message);
    }
  }

  frgpass() {
    // tslint:disable-next-line: whitespace
    if(this.email.invalid) {
     this.toast.info('Please Enter Email');
    // tslint:disable-next-line: one-line
    }else{
      this.service.aforgot({email: this.profileForm.value.email}).subscribe((res) => {
        this.addCookie(res.data.email);
        this.cookie.set('email', res.data.email);
        if (res){
          this.toast.success(res.sucess, res.message);
          this.router.navigate([`${'/aregister/' + res.data.email}`]);
        }else{
          this.toast.error(res.sucess, res.message);
        }
      });
    }
  }
  ngOnInit() {
  }

}
```

Plogin :--

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormBuilder } from '@angular/forms';
import {Validators} from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from '../../auth.service';
import { ToastrService } from 'ngx-toastr';
import { CookieService } from 'ngx-cookie-service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  profileForm = this.fb.group({
    email : ['', Validators.required],
    password : ['', Validators.required],
  });
  // tslint:disable-next-line: max-line-length
  constructor(private fb: FormBuilder, private router: Router, private service: AuthService, private toast: ToastrService, private cookie: CookieService) { }
```

```
get email() {return this.profileForm.get('email'); }
get password() {return this.profileForm.get('password'); }

addCookie(data: any) {
  this.cookie.set('Authorization', data );
}
getCookie() {
  return this.cookie.get('Authorization');
}

onSubmit() {
  if (this.profileForm.invalid) {
    this.profileForm.get('email').markAsTouched();
    this.profileForm.get('password').markAsTouched();
  // tslint:disable-next-line: one-line
  }else{
    this.service.loginUser({
      email: this.profileForm.value.email,
      password: this.profileForm.value.password
    }).subscribe((res) => {
      this.addCookie(res.token);
      this.cookie.set('id', res.data);
      this.showtoasterdf(res);
    });
  }
}
showtoasterdf(res) {
  // tslint:disable-next-line: whitespace
  if(res.sucess) {
    this.toast.success(res.sucess, res.message);
    this.router.navigate([`${'/dashboard'}`]);
  // tslint:disable-next-line: one-line
  }else{
    this.toast.warning(res.sucess, res.message);
  }
}

frgpass() {
  // tslint:disable-next-line: whitespace
  if(this.email.invalid) {
   this.toast.info('Please Enter Email');
  // tslint:disable-next-line: one-line
  }else{
    this.service.forgot({email: this.profileForm.value.email}).subscribe((res) => {
      this.addCookie(res.data.email);
```

```
        this.cookie.set('email', res.data.email);
        this.toast.success(res.sucess, res.message);
        this.router.navigate([`${'/register/'+ res.data.email}`]);
      });
    }
  }
  ngOnInit() {
  }

}
```

- ## Routing.js :--

Alogin:--

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AloginComponent } from './alogin.component';
import { AdashboardModule } from 'src/app/adashboard/adashboard.module';
import { AdashboardComponent } from 'src/app/adashboard/adashboard.component';


const routes: Routes = [
  { path: '', component: AloginComponent},
  { path: 'adashboard', component: AdashboardComponent}

];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class AloginRoutingModule { }
```

Plogin:--

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { LoginComponent } from './login.component';
import { DashboardComponent } from 'src/app/dashboard/dashboard.component';
```

```
const routes: Routes = [
  { path: '', component: LoginComponent},
  { path: 'dashboard', component: DashboardComponent}
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class LoginRoutingModule { }
```

Home:--

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home.component';
import { AregisterModule } from './aregister/aregister.module';
const routes: Routes = [
  { path : '', component : HomeComponent},
  { path: 'login', loadChildren: () => (import('./login/login.module').then(m => m.LoginModule))},
  { path: 'register', loadChildren: () => (import('./register/register.module').then(m => m.RegisterModule))},
  { path: 'register/:data', loadChildren: () => (import('./register/register.module').then(m => m.RegisterModule))},
  // tslint:disable-next-line: max-line-length
  { path: 'uforgotpassword', loadChildren: () => (import('./uforgotpassword/uforgotpassword.module').then(m => m.UforgotpasswordModule))},
  { path: 'aforgotpassword', loadChildren: () => (import('./aforgotpassword/aforgotpassword.module').then(m => m.AforgotpasswordModule))},
  { path: 'alogin', loadChildren: () => (import('./alogin/alogin.module').then(m => m.AloginModule))},
  { path: 'aregister', loadChildren: () => (import('./aregister/aregister.module').then(m => AregisterModule))},
  { path: 'aregister/:data', loadChildren: () => (import('./aregister/aregister.module').then(m => AregisterModule))},
  { path: 'about', loadChildren: () => (import('./about/about.module').then(m => m.AboutModule))}
];
@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class HomeRoutingModule { }
```

TEST CASE DESIGN

Thorough testing is required to run the project at client end successfully. I prepared a number of test cases and input data from the developed project and saw their impact in the final reports for example.

Test Case I

All the fields are entered with the correct values in the Registration form on the website. On clicking Sign up, all the data must be stored in Accounts table. The test is successful.

Test Case II

In the login form, entering correct username with wrong password must display the message for writing correct credentials. The test is successful.

Test Case III

In the user panel, searching the webpage with two letters must display all the webpages containing those two letters together in the heading. The test is successful.

Test Case IV

In the user panel, while setting an alert, if the date or time is being set for the expired time, an error must pop-up displaying to choose correct time/date. The test is successful.

TESTING

System Testing

Software testing is a critical element of software quality assurance and represent the ultimate review of specification, design, coding. The purpose of product testing is to verify and validate the various work products viz. units, integrated unit, final product to ensure that they meet their requirements. In software development phase and error can be injected at any stage during development, so, the different type of testing is required on different levels. For using all levels of testing which are essential to test client needs, requirements, design and coding, we have to follow different type of methods of testing to test the different stages of development. Software Testing is the process of testing the software project. Effective software will contribute to the delivery of higher quality software products, more satisfied users, lower maintenance costs, more accurate, and reliable results.

Testing Methodologies

Testing presents an interesting anomaly. A series of test cases are created that are created intended to “demolish” the software that has been created. Testing requires that the developer discard preconceived nations of the “correctness” of the software just developed and overcome a conflict of interest that occurs when errors are uncovered.

Testing Objectives

These are as follows :-

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding undiscovered errors.
- A successful test case is one that uncovers yet undiscovered error.

Our objective is to design test cases that systematically uncover different classes of errors and do so with a minimum amount of time and effort.

This process has two parts:

Planning: This involves writing and reviewing unit, integration, functional, validation and acceptance test plans.

Execution: This involves executing these test plans, measuring, collecting data and verifying if it meets the quality criteria. Data collected is used to make appropriate changes in the plans related to development and testing.

The quality of a product or item can be achieved by ensuring that the product meets the requirements by planning and conducting the following tests at various stages.

Verification and Validation

Software testing is used in association with verification and validation:

- Verification: Have we built the software right (i.e., does it match the specification?)? It is process based.
- Validation: Have we built the right software (i.e., is this what the Patient wants?)? It is product based.

TYPES OF TESTING

The main types of software testing are:

Unit Testing:

Starting from the bottom the first test level is Unit Testing (Component testing). It involves checking that each feature specified in the “Component Design” has been implemented in the component. In theory an independent tester should do this, but in practice the developer usually does it, as they are the only people who understand how a component works. The problem with a component is that it performs only a small part of the functionality of a system, and it relies on co-operating with other parts of the system, which may not have been built yet. To overcome this, the developer either builds, or uses special software to trick the component into believing it is working in a fully functional system.

Interface Testing:

As the components are constructed and tested they are then linked together to check if they work with each other. It is fact that two components that have passed all their tests, when connected to each other produce one new component which may be full of faults. These tests can be done by specialists, or by the developers. Interface testing is not focused on what the components are doing but on how they communicate with each other. The tests are organized to check all the interfaces, until all the components have been built and interfaced to each other producing the whole system.

System Testing:

Once the entire system has been built then it has to be tested against the "System Specification" to check if it delivers the features required. It is still developer focused, although specialist developers known as system testers are normally employed to do it. In essence System testing is not about checking the individual parts of the design, but about checking the system as a whole. In effect it is one giant component. System testing can involve a number of specialist types of test to see if all the functional and non-functional requirements have been met. In addition to functional requirements these may include the following types of testing for the non-functional requirements:

1). Performance- Are the performance criteria met?

2). Volume- Can large volumes of information be handled?

3). Stress- Can peak volumes of information be handled?

4). Documentation- Is the documentation usable for the system?

5). Robustness- Does the system remain stable under adverse circumstances?

Acceptance Testing:

Acceptance testing checks the system against the "Requirements". It is similar to system testing in that the whole system is checked but the important difference is the change in focus: System testing checks that the system that was specified has been delivered. Acceptance testing checks that the system delivers what was requested. The Patient and not the developer should always do acceptance testing. The Patient knows what is required from the system to achieve value in the business and is the only person qualified to make that judgment. The forms of tests may follow those in system testing, but at all times they are informed by the business needs.

Release Testing:

Even if a system meets all its requirements, there is still a case to be answered that will it benefit the business? Release testing is about seeing if the new or changed system will work in the existing business environment. Mainly this means the technical environment, and checks concerns such as:

- Does it affect any other systems running on the hardware?
- Is it compatible with other system?
- Does it have acceptable performance under load?

TEST CASE DESIGN

Test case design focuses on a set of techniques for the creation of test cases that meet overall testing objectives. In test case design phase, the engineer creates a series of test cases that are intended to “demolish” the software that has been built.

Any software product can be tested in one of two ways:

1). Knowing the specific function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational, at the same time searching for errors in each function. This approach is known as Black Box Testing.

2). Knowing the internal workings of a product, tests can be conducted to ensure that internal operation performs according to specifications and all internal components have been adequately exercised. This approach is known as White Box Testing.

Black box testing:

Black box testing is designed to uncover errors. They are used to demonstrate that software functions are operations; that input is properly accepted and output is correctly produced; and that integrity of external information is maintained. A black box examines some fundamental aspects of a system with little regard for the internal logical structure of the software.

White box testing:

White box testing of software is predicated on close examination of procedural details. Providing test cases that exercises specific set of conditions and/or loops, tests logical paths through the software. The “state of program” may be examined at various points to determine if the expected or asserted status corresponding to the actual status.

DEBUGGING AND CODE IMPROVEMENT

In ideal worlds, all programmers would be so skilled and attentive to detail that they would write bug-free code. Unfortunately, we do not live in an ideal world. As such, debugging, or tracking down the source of errors and erroneous result, is an important task that all developers need to perform before they allow end-user to use their applications. Some techniques for reducing the number of bugs in code up front. There are three categories of bugs:

Syntax error:

These errors occur when code breaks the rule of the language, such as a forgotten closing curly braces ({}) in c#. This error is the easiest to locate. The language complier or integrated development environment (IDE) will alert the coder to them and will not allow to compile the program until it is corrected.

Semantic error:

These errors occur in code that is correct according to rules of the compiler, but that causes unexpected problems such as crashes or hanging on execution. A good example is code that execute in a loop but never exists the loop, either because the loop depends on the variable whose values was expected to be something different than it actually was or because the programmer forget to increment the loop counter. Another category of errors in this area includes requesting a field from a dataset, there is no way to tell if the field actually exists at compile time. These bugs are harder to detect and are one type of running error.

Logic error:

Logic errors are like semantic errors, logic errors are runtime error. That is, they occur while the program is running. But unlike semantic errors, logic errors do not cause the application to crash or hang. Logic error results in unexpected values or output. This can be a result of something as simple as a mistyped variables name that happens to match another declared variable in the program. This type of error can be extremely difficult to track down to eliminate.

Preventing Debug Write readable code

Develop and make consistent use of naming and coding standards. It not that important which standard we use, such as Hungarian notation or Pascal, Casing (First Name) or other naming conventions, as long as we use one. We should also strive for consistency in our comments and encourage liberal commenting code.

Create effective test plan

The only effective way to eliminate logic error is to test very path of your application with every possible data values that a user could enter. This is difficult to manage without effective planning. We should create our test plan at the same time we are designing the application, and we should update these plans as you modify the application design

SYSTEM SECURITY MEASURES

Computer systems face a number of security threats. One of the basic threats is data loss, which means that parts of a database can no longer be retrieved. This could be the result of physical damage to the storage medium (like fire or water damage), human error or hardware failures.

Another security threat is unauthorized access. Many computer systems contain sensitive information, and it could be very harmful if it were to fall in the wrong hands. Imagine someone getting a hold of your social security number, date of birth, address and bank information.

Getting unauthorized access to computer systems is known as hacking. Computer hackers have developed sophisticated methods to obtain data from databases, which they may use for personal gain or to harm others.A third category of security threats consists of viruses and other harmful programs. A computer virus is a computer program that can cause damage to a computer's software, hardware or data. Itis referred to as a virus because it has the capability to replicate itself and hide inside othercomputer files.

The objective of system security is the protection of information and property from theft, corruption and other types of damage, while allowing the information and property to remain accessible and productive. System security includes the development and implementation of security countermeasures.

The system security problem can be divided into four related issues-

- Security
- Integrity
- Privacy
- Confidentiality

System security:

Refers to the technical innovations and procedures applied to the hardware and operating systems to protect against deliberate or accidental damage from a defined threat. In contrast data security is the protection against data from loss disclosure, modification and destruction.

System integrity:

Refers to the proper functioning of hardware and programs, appropriate physical security, and safe against external threats such as eavesdropping and wire-tapping. In contrast data integrity make sure that data do not differ from their original form and have not been accidentally or intentionally disclosed, altered or destroyed.

Privacy:

Defines the rights of the users or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

Confidentiality:

Confidentiality is a special status given to sensitive information in a database to minimize possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

Control Measures:

After system security risks have been evaluated, the next step is to select the measures that are internal and external to the facility. The measures are generally classified under the following:

Access Control: Various steps are taken to control access to a computer facility. One way is to use an encoded card system with a log-keeping capability. Encryption is an effective and practical way to safeguard data transmitted over an unprotected communications channel.

Audit Controls: It protects a system from external security breaches and internal fraud or embezzlement. The resources invested in audit controls, however should balance with the sensitivity of the data being manipulated. One problem with audit controls is that it is difficult to prove their worth until the system has been violated or a company officer imprisoned. For this reason, audibility must be supported at all management levels and planned into every system.

DATABASE SECURITY

Much attention has been focused on network attacks by crackers, and how to stop these. But the vulnerability of data inside the database is somewhat overlooked. Databases are far too critical to be left unsecured or incorrectly secured.

Most companies solely implement perimeter-based security solutions, even though the greatest threats are from internal sources. And information is more often the target of the attack than network resources.

Securing a database allows organizations to protect the corporate data from threats from external sources. Database Security is a serious issue, and if not implemented correctly, the consequences can be costly to companies if their vital data is hacked into, or their Patient's data "leaks" out which can even lead to cases of identity theft.

The best security practices protect sensitive data as it's transferred over the network (including internal networks) and when it's at rest. One option for accomplishing this protection is to selectively parse data after the secure communication is terminated and encrypt sensitive data elements at the SSL/Web layer. Doing so allows enterprises to choose at a very granular level (usernames, passwords, and so on.) the sensitive data to secure throughout the enterprise.

Application-layer encryption and mature database-layer encryption solutions allow enterprises to selectively encrypt granular data into a format that can easily be passed between applications and databases without changing the data.

Data Encryption

The sooner data encryption occurs, the more secure the information is. Due to distributed business logic in application and database environments, organizations must be able to encrypt and decrypt data at different points in the network and at different system layers, including the database layer. Encryption performed by the DBMS can protect data at rest, but you must decide if you also require protection for data while it‟s moving between the applications and the database and between different applications and data stores. Sending sensitive information over the Internet or within your corporate network as clear text defeats the point of encrypting the text in the database to provide data privacy.

SECURITY IN THIS PROJECT

In the website, there are two types of entities. One is the user and the other is the Deoctor. Different types of permissions are given to them according to their rights. Only authorized users can use the services of the website. The user can access the services by providing the valid username and password. Each and every user has their personal account. No user can access some other user’s account. The user can view only his saved webpage, notes and alerts. Access to the personal account is granted only to the users with valid Username and Password in the login page. Otherwise access to the personal account is denied.

Choosing of right codes to minimize number of lines of code and as result to take less execution time. Capable of handling all kinds of inter-actions and inter-dependencies of the modules especially when used in a true multi-user mode. Creation of sessions on login for each user which will expire as soon as they logout. Easily modifiable to rectify and type of errors found during testing or actual operations. Provision to include new requirements or modules easily and also for module deletions. Inclusion of entire new modules becomes easy and simple.

CONCLUSION

The project Medi-Info will provide flexibility to the Patient as they can access the desired news whenever they want. There is no limitation of place and medium. This makes news device independent and can be accessed anywhere in the world, on any system that has access to the Internet.

As all the medicine are available , this provides reliability to the Patient. Since all the medicine are available at the single platform , Patient is not required to search them on multiple websites and thus news are more organized making it easy to manage. Hence it saves user's time and effort.

FUTURE SCOPE AND FUTHER ENHANCEMENT

FUTURE SCOPE

The proposed website can be useful to the Patient and for the Doctor's , who are fond of reading daily use medicine for particular disease , and also the Doctor who are preparing for the completive medicine set.

This website can be enhanced by including the modules, which are related to other sectors. The website can also be used by Educational institutions where all the students and teachers can register and read news, related to the educational category with each other without bothering about uploading the content and its data size.

FURTHER ENHANCEMENT

The website proposes some enhancements which can be implied in future:

 Further New Categories will be added.

 Another different medicine will be available on a platform.

 Support for Pop-Up alerts.

 Site policies will be established.

 Better database management.

 Implementation of new database technology.