

**DOCKER BASED DECENTRALIZED LOGISTICS  
MANAGEMENT SYSTEM**

**A PROJECT REPORT**

*Submitted by*

**NARESH KUMAR K**

**111616106082**

*In partial fulfillment for the award of the degree*

*Of*

**BACHELOR OF ENGINEERING**

**in**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**RMK COLLEGE OF ENGINEERING AND TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600025**

**APRIL 2020**

**ANNA UNIVERSITY: CHENNAI 600025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**DOCKER BASED DECENTRALIZED LOGISTICS MANAGEMENT SYSTEM**” is the bonafide work of “**NARESH KUMAR K**” who carried out the project work under my supervision

**SIGNATURE**

**Dr.N.GANGATHARAN**

**HEAD OF THE DEPARTMENT**

Professor

Department of Electronics and  
Communication Engineering

RMK College of Engineering and  
Technology,

RSM Nagar Puduvoyal,

Chennai – 601206.

**SIGNATURE**

**MS.BHARATHA SREEJA G**

**SUPERVISOR**

Assistant Professor

Department of Electronics and  
Communication Engineering

RMK College of Engineering and  
Technology,

RSM Nagar Puduvoyal,

Chennai – 601206.

**Submitted for the project examination held on \_\_\_\_\_**

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

Support on demand encouragement at the needed moment and guidance in the right direction are indispensable for the success of any project. I have received these in excess from all corners from various people, I am glad to submit my gratitude to them.

I thank **Shri.R.S.Munirathinam**, chairman and **Shri.R.M.Kishore**, Vice chairman of RMK group of Institution for extending a generous hand in providing the best of resources to the college **Dr.T.Rengaraja**, Principal has been a source of motivation to all the staffs and students of my college.

My sincere thanks to **Dr.N.Gangatharan**, Head of the Department for his continuous support and motivation throughout the project

I extend my profound gratitude to **Dr.C.Arun**, my project coordinator, **Mr.Srinivasan Thanukrishnan**, CEO, Glosys Technology Solutions Pvt. Ltd, my project guide for his guidance, who has been a polestar throughout the course of the project, I thank **Ms.G.Bharatha Sreeja**, my project supervisor for giving her support to complete the project successfully.

Last, but not least, we take this opportunity to thank all the staff members of the Department of Electronics and Communication Engineering. Regards to my family, my classmates and friends who offered and unflinching moral support for completion of this project.

# **DOCKER BASED DECENTRALIZED LOGISTICS MANAGEMENT SYSTEM**

## **ABSTRACT**

Nowadays we buy more of our vital stuff online than offline and for our transactional purposes we make use of banks, UPI vendors as our third party to regulate the virtual flow of our money. These third parties use various security algorithms such as [RSA ] for making their virtual transactions more secure, This Decentralized transactional approach makes the virtual transactions to occur without any third parties such as banks, UPI vendors by making use of the advancements in web application development and the block chain technology. These features are added to the e-commerce application to form a complete product rather than being a simple experiment. Then the app's middleware is deployed to servers as the docker container, it optimizes the memory usage of the application and monitor the performance of the application which helps in preventing overloading the servers

## **TABLE OF CONTENTS**

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE. NO</b>
	ACKNOWLEDGEMENT	3
	ABSTRACT	4
	LIST OF FIGURES	9
<b>1</b>	<b>INTRODUCTION</b>	10
	1.1 PURPOSE	10
	1.2 PROBLEM STATEMENT	11
	1.3 OBJECTIVE	11
	1.4 EXISTING SYSTEM	11
	1.5 PROPOSED SYSTEM	12
	1.5.1 DISADVANTAGE OF EXISTING SYSTEM	12
	1.5.2 ADVANTAGES OF PROPOSED SYSTEM	13

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>14</b>
<b>3</b>	<b>INTRODUCTION TO WEB APPLICATION</b>	
	<b>DEVELOPMENT ( SOFTWARE USED )</b>	<b>24</b>
	<b>3.1 SYSTEM REQUIREMENTS</b>	<b>24</b>
	3.1.1 DEVELOPMENTAL REQUIREMENTS	24
	3.1.2 END USER REQUIREMENTS	26
	<b>3.2 INTRODUCTION TO FRONTEND</b>	
	<b>DEVELOPMENT</b>	<b>26</b>
	3.2.1 INTRODUCTION TO HTML	26
	3.2.2 INTRODUCTION TO CSS	27
	3.2.3 INTRODUCTION TO JS AND TS	28
	<b>3.3 INTRODUCTION TO ANGULAR</b>	<b>29</b>
	3.3.1 FOLDER STRUCTURE	31

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE. NO</b>
	<b>3.4 INTRODUCTION TO MIDDLEWARE</b>	<b>32</b>
	3.4.1 EXPRESS MIDDLEWARE	32
	3.4.2 FLASK MIDDLEWARE	33
	<b>3.5 INTRODUCTION TO MONGODB</b>	<b>33</b>
	<b>3.6 INTRODUCTION TO RESTAPI</b>	<b>34</b>
	<b>3.7 INTRODUCTION TO ETHEREUM</b>	<b>35</b>
<b>4</b>	<b>DOCKER BASED DECENTRALIZED LOGISTIC</b>	
	<b>MANAGEMENT SYSTEM</b>	<b>36</b>
	4.1 FRONTEND OPERATION	36
	4.2 MIDDLEWARE OPERATION	36
	4.3 DATABASE	37
	4.4 ETHEREUM NETWORK	37
	4.4 SYSTEM DESIGN	38
	4.4 ARCHITECTURAL DESIGN	39

<b>5</b>	<b>RESULTS AND DISCUSSION</b>	<b>40</b>
<b>6</b>	<b>CONCLUSION</b>	<b>43</b>
<b>7</b>	<b>REFERENCES</b>	<b>44</b>



## **LIST OF FIGURES**

<b>FIG NO</b>	<b>TITLE</b>	<b>PAGE</b>
3.1	ANGULAR ARCHITECTURE	29
3.2	MODULE STRUCTURE	31
3.3	MIDDLEWARE INIT EXPRESS	32
3.4	MIDDLEWARE INIT FLASK	33
4.1	SYSTEM DESIGN	38
4.2	ARCHITECTURAL DESIGN	39
5.1	REGISTRATION PAGE	40
5.2	FORGOT PASSWORD PAGE	40
5.3	PRODUCT PAGE	41
5.4	SELLER ORDERS PAGE	41
5.5	USER PROFILE	42
5.6	PRODUCT REGISTRATION PAGE	42

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 PURPOSE**

Online shopping is one of the common indispensable unsolvable problems in the modern world. As E-commerce has an ever growing demand in the web stack development, in the early models it has been implemented as a static sites that merely have a payment option with limited stocks and does not employ the product tracking facilities but due to the significant growth in the web technologies and with the help of arrival new languages that support concurrency and the frameworks that allow the development and testing process simple, this makes the product development cycle faster than ever before. With the help of this developments in the web technologies and language that support multithreading functionality I have used the advanced web frameworks. In this project I have focused on solving the problems related to the monetary transactions and the costs required to ship the ordered products from the seller to customer of the total e-commerce application. The monetary transaction were made to take place without the need of the third parties such as banks, UPI vendors. This is possible by the advancements in the web stack development and with arrival of new technology of block chain, they use data consensus primarily to make this transactions successful. The shipping costs of the products have been calculated by using transportation optimizing models. These models serve a great purpose in optimizing the cost.

## **1.2 PROBLEM STATEMENT**

It is known that our monetary transactions are taken care by third parties such as banks and UPI vendors, they inevitable do some of the charges for our monetary transactions design a system that helps to eliminate the use of third parties such as banks and UPI vendors and establish an effective way of monetary transaction. Also estimate the optimized way for the mobility of products for the logistics of the application.

## **1.3 OBJECTIVE**

The objective are more vital for designing the system that enables the current disability and introduces a newer way for the usage of the application. Here are primary objectives are

- To develop a decentralized web application
- To optimize the costs in the logistics
- To improve the security in the monetary transactions
- To eliminate the involvement of third parties

These are the primary objectives that are kept in clear in development of the product/project. These objectives guides me through the development of the project.

## **1.4 EXISTING SYSTEM**

The existing systems of the e-commerce have one or middle wares that are developed in the languages that are well tested and have advanced concepts like concurrency and multithreading are patched later and these environments have slower development cycles. Existing systems have JAVA as the primary environment as their middleware and SQL as their primary transactional database because they are hard tested and are less vulnerable to threats. Some of them still use website approach rather than a SPA approach.

## **1.5 PROPOSED SYSTEM**

In my proposed systems I have developed the application in a micro services architecture. It has more than one middleware each one has the environment specialized for their purpose rather than relying on one middleware for all purpose. In my proposed system I have used Angular for creating a Single Page Application – a Progressive Web App that runs well on cross platforms such as Android, iOS, etc. I have used python flask as the middleware for the optimization of the logistics and Nodejs Express as the middleware for the registrations, authentications and making transactions with the block chain. Since JavaScript runs asynchronously it effectively uses the runtime thus improves the performance of the system.

### **1.5.1 DISADVANTAGE OF EXISTING SYSTEM**

Since existing systems use older middleware it could potentially have some of the issues with reactive programming paradigm since it could not operate asynchronously as the modern languages and environments do. Here are the important demerits to consider.

- Partially supports multithreading
- Lacks in reactive paradigm
- The need for the third parties such as banks, etc. are necessary

### **1.5.2 ADVANTAGES OF PROPOSED SYSTEM**

Merits of the proposed systems should overcome the demerits of the existing system to improve the abilities of the existing system. The proposed system overcomes the demerits mentioned above and thereby providing primary advantages for the proposed systems. Here are the primary merits of the proposed systems.

- Has reactive programming paradigm
- Has the multithreading runtime environment
- The system also eliminates the need for the third parties such as banks,etc.

These merits can be achieved by using the recently developed frameworks such as Angular and Expressjs, they use the modern reactive programming paradigms to provide rich UI/UX.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **Big data analytics and application for logistics and supply chain management**

**Author:** Kannan Govindan

**YEAR 2018**

This special issue explores big data analytics and applications for logistics and supply chain management by examining novel methods, practices, and opportunities. The articles present and analyse a variety of opportunities to improve big data analytics and applications for logistics and supply chain management, such as those through exploring technology-driven tracking strategies, financial performance relations with data driven supply chains, and implementation issues and supply chain capability maturity with big data. This editorial note summarizes the discussions on the big data attributes, on effective practices for implementation, and on evaluation and implementation methods.

#### **Synchromodal logistics: An overview of critical success factors, enabling technologies, and open research issues**

**Author:** RiccardoGiusti; DanieleManerba; GiorgioBruno; RobertoTadei;

**YEAR: 2020**

As supply chain management is becoming demand driven, logistics service providers need to use real-time information efficiently and integrate new technologies into their business. Synchromodal logistics has emerged recently to improve flexibility in supply chains, cooperation among stakeholders, and utilization of resources. We survey the existing scientific literature and real-life developments on synchromodality. We focus on the critical success factors of synchromodality and six categories of enabling technologies. We identify open research issues and propose the introduction of a new stakeholder, which takes on the role of orchestrator to coordinate and provide services through a technology-based platform.

## **Blockchain-technology-supported platforms for diamond authentication and certification in luxury supply chains**

**Author:** Tsan-Ming Choi;

**YEAR 2019**

The blockchain technology is very useful in many industries. One current application is on diamond authentication and certification, which is important in many luxury supply chains. In this paper, we explore different consumer utility driven operations models and highlight the values of blockchain technology supported (BTS) platforms for diamond authentication and certification. We build models and analytically examine both the traditional retail network operations (Model R) and the BTS selling platform (Model PL). We further extend the analysis to study the case with the BTS certification platform (Model BCR). We reveal the conditions under which one model outperforms the others. In particular, we note that the shopping convenience utility offered by the traditional retailers is a critical factor determining which model is the best. Finally, for the BTS platform operations, we study the blockchain-technology-based diamond authentication and certification (BDAC) cost and reveal that reducing it is beneficial to all parties in the luxury supply chain.

## **Peer-to-peer collaborative consumption for fashion products in the sharing economy: Platform operations**

**Author:**Choi, Tsan-Ming; He, Yanyan

**YEAR 2019**

With the advance of platform technologies, peer-to-peer collaborative consumption (P2P-CC) is getting very popular for fashion products in the sharing economy. With this arrangement, one important implication to consumers is that the fashion product's value seems to be larger with P2P-CC than without. Motivated by the observed industrial practice on P2P-CC for fashion products, we construct stylized models and conduct an analytical study in this paper. We analytically prove that comparing between the operations with and without P2P-CC, the presence of P2P-CC always benefits the fashion brand (i.e. the firm) as well as the consumers

who buy the product. For the relative benefits brought by P2P-CC between the firm and consumers, we prove that the firm's expected profit is double compared to the consumers under the case without P2P-CC. However, this proportion is smaller under the case with P2P-CC. Regarding the platform, we find that it is beneficial for the platform to adopt the revenue sharing scheme, instead of the fixed service charging scheme, even though adopting either one does not change the qualitative conclusion on the benefits of P2P-CC. To check the robustness of results, we extend the model and conduct further analysis to highlight the impacts brought by the fashion brand's advertisement decisions, the presence of different consumer segments, and the probable service delivery charge. We uncover that the main derived conclusion remains true in all these extensions. Managerial implications are then discussed.

## **Maritime shipping digitalization: Blockchain-based technology applications, future improvements, and intention to use**

**Author:** Yang, Chung-Shan

**Year** 2019

Although a number of studies focus on using the blockchain technology (BT) in various application aspects, there is no comprehensive survey on the blockchain applications in maritime shipping supply chain. To fill this gap, we conduct a comprehensive blockchain applications and future improvements survey, and empirically evaluate its effects on intention to use. The results suggested that customs clearance and management, digitalizing and easing paperwork, standardization and platform development dimensions positively affected intention to use. In particular, this paper gives the maritime shipping blockchain-based digitalization also points out the future improvement directions in the blockchain technology.



## Angular 9

**Authors:**Alan Agius ; Alex Rickabaugh ; Andrew Kushnir ; Andrew Scott ; Andrew Seguin ; Charles Lyding ; Dave Shevitz ; Denny Brown

**YEAR: 2020**

Angular 9 was released on February 7, 2020. Version 9 moves all applications to use the Ivy compiler and runtime by default. Angular has been updated to work with TypeScript 3.6 and 3.7. In addition to hundreds of bug fixes, the Ivy compiler

and runtime offers numerous advantages:

- Smaller bundle sizes
- Faster testing
- Better debugging
- Improved CSS class and style binding
- Improved type checking
- Improved build errors
- Improved build times, enabling AOT on by default
- Improved Internationalization

## Expressjs

**Authors:**TJ Holowaychuk ; StrongLoop

**YEAR: 2019**

**Express.js**, or simply **Express**, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

The author, TJ Holowaychuk, described it as a Sinatra-inspired server, meaning that it is relatively minimal with many features available as plugins. Express is the back-end component of the MEAN stack, together with the MongoDB database software and Angular front-end framework.

## **ETHEREUM**

**Author:** Vitalik Buterin ; Gavin Wood

**YEAR: 2019**

Ethereum is an open source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality. It supports a modified version of Nakamoto consensus via transaction-based state transitions.

Ether is a fundamental token for operation of Ethereum, which thereby provides a public distributed ledger for transactions. It is used to pay for gas, a unit of computation used in transactions and other state transitions. Mistakenly, this currency is also referred to as Ethereum.

## **FLASK**

**Author:** Armin Ronacher

**YEAR: 2019**

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.[3] It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

## **REST**

**Author:** Roy Fielding

**YEAR:** 2000

Representational state transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the Internet. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations

Web resources were first defined on the World Wide Web as documents or files identified by their URLs. However, today they have a much more generic and abstract definition that encompasses every thing or entity that can be identified, named, addressed, or handled, in any way whatsoever, on the Web. In a RESTful Web service, requests made to a resource's URI will elicit a response with a payload formatted in HTML, XML, JSON, or some other format. The response can confirm that some alteration has been made to the stored resource, and the response can provide hypertext links to other related resources or collections of resources. When HTTP is used, the operations (HTTP methods) available are GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS and TRACE.

## **JAVASCRIPT**

**Author:** Brendan Eich

**YEAR:** 2019

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast

majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). However, the language itself does not include any input/output (I/O), such as networking, storage, or graphics facilities, as the host environment (usually a web browser) provides those APIs.

JavaScript engines were originally used only in web browsers, but they are now embedded in server-side website deployments, usually via Node.js.

## **NODEJS**

**Author:** Ryan Dahl

**YEAR: 2019**

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server- and client-side scripts.

Though .js is the standard filename extension for JavaScript code, the name "Node.js" doesn't refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications

## **TYPESCRIPT**

**Author:** Microsoft

**YEAR: 2019**

TypeScript is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript, and adds optional static typing to the language.

TypeScript is designed for development of large applications and transcompiles to JavaScript. As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs. TypeScript may be used to develop JavaScript applications for both client-side and server-side execution (as with Node.js or Deno).

There are multiple options available for transcompilation. Either the default TypeScript Checker can be used, or the Babel compiler can be invoked to convert TypeScript to JavaScript.

## **FIREBASE**

**Author:** James Tamplin, Andrew Lee

**YEAR: 2018**

Firebase evolved from Envolv, a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolv provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that were not chat messages. Developers were using Envolv to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in September 2011 and it launched to the public in April 2012.

Firebase's first product was the Firebase Real-time Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud. The product assists software developers in building real-time, collaborative applications

## **DOCKER**

**Author:** Solomon Hykes ; Sebastien Pahl

**YEAR: 2018**

Hykes started the Docker project in France as an internal project within dotCloud, a platform-as-a-service company. Docker debuted to the public in Santa Clara at PyCon in 2013. It was released as open-source in March 2013. At the time, it used LXC as its default execution environment. One year later, with the release of version 0.9, Docker replaced LXC with its own component, which was written in the Go programming language.

## **VISUAL STUDIO CODE**

**Author:** Microsoft

**YEAR: 2015**

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The source code is free and open source and released under the permissive MIT License. The compiled binaries are freeware and free for private or commercial use.

Visual Studio Code is based on Electron, a framework which is used to develop Node.js applications for the desktop running on the Blink layout engine

## **GITHUB**

**Author:** Tom Preston-Werner ; Chris Wanstrath ; P. J. Hyett ; Scott Chacon

**YEAR: 2008**

GitHub, Inc. is a US-based global company that provides hosting for software development version control using Git. It is a subsidiary of Microsoft, which acquired the company in 2018 for US\$7.5 billion. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

GitHub offers plans free of charge, and professional and enterprise accounts. Free GitHub accounts are commonly used to host open source projects. As of January 2019, GitHub offers unlimited private repositories to all plans, including free accounts. As of January 2020, GitHub reports having over 40 million users and more than 100 million repositories (including at least 28 million public repositories), making it the largest host of source code in the world.

## **GIT**

**Author:** Linus Torvalds

**YEAR: 2005**

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano. As with most other distributed version-control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server.

## **CHAPTER 3**

### **INTRODUCTION TO WEB APPLICATION DEVELOPMENT**

A web application is a software application that runs on a remote server. In most cases, Web browsers are used to access Web applications, over a network, such as the Internet. Some web applications are used in intranets, in companies and schools, for example. Web applications are different from other applications because they do not need to be installed.

Some example web applications are: Facebook (social networking), Flickr (photo sharing) and Mibbit (chatting)

Web applications are popular because most computer operating systems have web browsers. Programmers can easily change a web application. Users do not need to install any new software to see these changes.

Here in my project the complete stack is built by

- Angular frontend application
- Express middleware application
- Flask middleware application

### **3.1 SYSTEM REQUIREMENTS**

The system requirements are necessary for both the developmental and usage purposes, during the developmental stage the system requirements are needed more than at the realtime usage of application

#### **3.1.1 DEVELOPMENTAL REQUIRMENTS**

The development cycle of the application requires bit more hardware and softwares for developing



### **HARDWARE REQUIRED FOR THE DEVELOPMENT:**

CPU Type	:	Intel core i3
Clock Speed	:	>2.0 GHz
RAM Size	:	>8 GB
Hard disk capacity	:	>100 GB
Monitor type	:	15.6 inch color monitor

### **SOFTWARE REQUIRED FOR THE DEVELOPMENT:**

Operating System	:	Win10/Ubuntu18.04/macOS 10.14
Runtime Environment	:	Nodejs
Framework	:	Expressjs, Angular, Flask
Version Control Tool	:	Git and Github
IDE	:	Visual Studio Code
Database	:	MongoDB

### **3.1.2 END USER REQUIREMENTS:**

Since the realtime usage does not require large amount to services to be run on the system, it consumes lesser RAM and requires minimum CPU time. This application was developed by making it run cross platform. It can able to run on any Andriod or iOS mobile device and in any desktops and laptops. This application requires a browser to use their runtime.

## **3.2 INTRODUCTION TO FRONTEND DEVELOPMENT**

Front end of the application is the key element to handles the inputs from the user and provides rich UI/UX to the user, it should be developed in a appropriate manner to satisfy both the requirements. Frontend application development is based on three different technologies such as

- HTML
- CSS
- JS or TS

### **3.2.1 INTRODUCTION TO HTML**

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behavior (JavaScript).

"Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.

HTML uses "markup" to annotate text, images, and other content for display in a Web browser. HTML markup includes special "elements" such as

<head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, <span>, <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, <li>

An HTML element is set off from other text in a document by "tags", which consist of the element name surrounded by "<" and ">". The name of an element inside a tag is case insensitive. That is, it can be written in uppercase, lowercase, or a mixture. For example, the <title> tag can be written as <Title>, <TITLE>, or in any other way.

### **3.2.2 INTRODUCTION TO CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

## **USAGE OF CSS:**

### **(i) EXTERNAL CSS**

The styles can also be placed in an external CSS file, as described below, and loaded using syntax similar to:

```
<link href="path/to/file.css" rel="stylesheet" type="text/css">
```

This further decouples the styling from the HTML document, and makes it possible to restyle multiple documents by simply editing a shared external CSS file.

## **3.2.3 INTRODUCTION TO JS AND TS**

### **(i) JAVASCRIPT**

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. Read more about JavaScript.

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about APIs specific to Web pages, please see Web APIs and DOM.

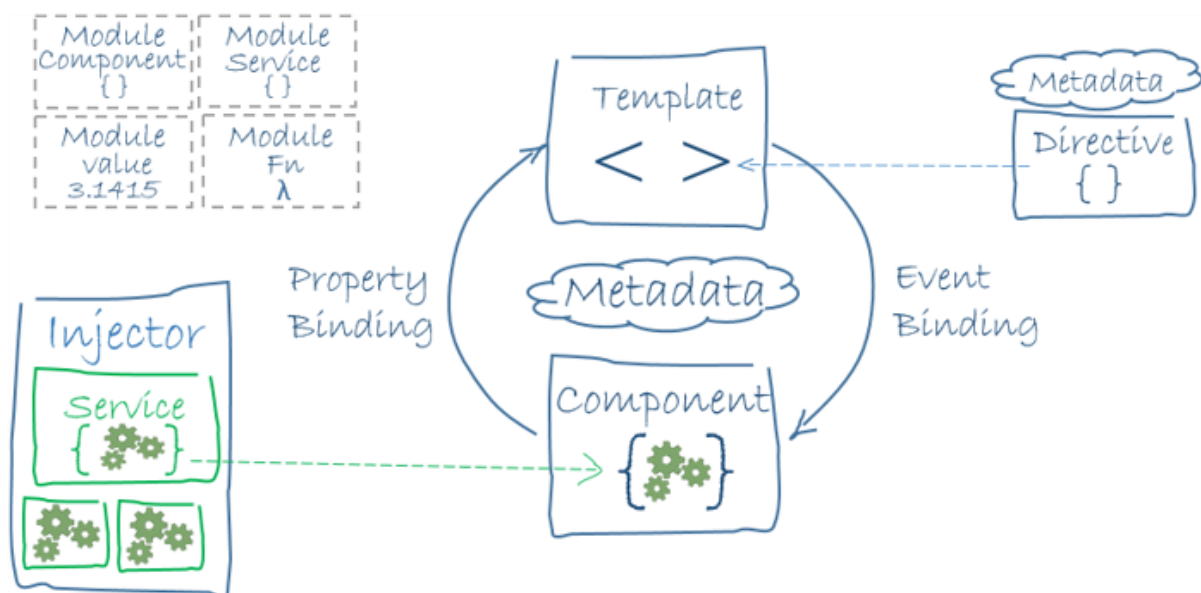
The standard for JavaScript is ECMAScript. As of 2012, all modern browsers fully support ECMAScript 5.1. Older browsers support at least ECMAScript 3. On June 17, 2015, ECMA International published the sixth major version of ECMAScript, which is officially called ECMAScript 2015, and was initially referred to as ECMAScript 6 or ES6. Since then, ECMAScript standards are on yearly release cycles. This documentation refers to the latest draft version, which is currently ECMAScript 2020.

## (ii) TypeScript

TypeScript is designed for development of large applications and transcompiles to JavaScript. As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs. TypeScript may be used to develop JavaScript applications for both client-side and server-side execution

## 3.3 INTRODUCTION TO ANGULAR

AngularJS is based on the model view controller, whereas Angular 2 is based on the components structure. Angular works on the same structure as Angular2 but is faster when compared to Angular2.



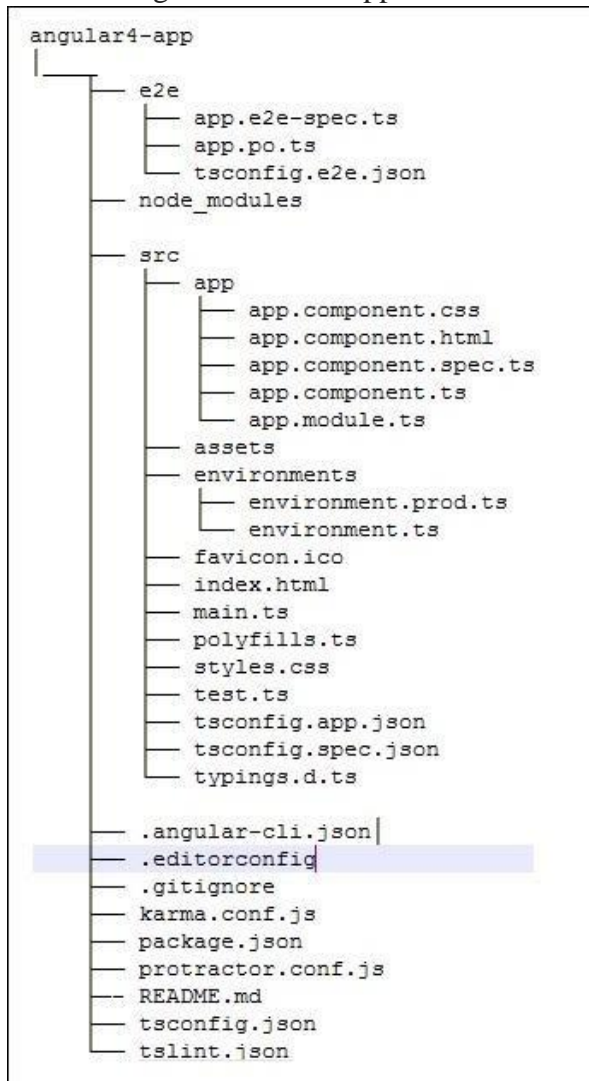
**Fig 3.1 Angular Architecture**

Angular uses TypeScript 2.2 version whereas Angular 2 uses TypeScript version 1.8. This brings a lot of difference in the performance.

To install Angular, the Angular team came up with Angular CLI which eases the installation. You need to run through a few commands to install Angular. Go to this site <https://cli.angular.io> to install Angular CLI.

### 3.3.1 FOLDER STRUCTURE

The Angular 4 app folder has the following folder structure



**Fig 3.2 Module Structure**

**e2e** – end to end test folder. Mainly e2e is used for integration testing and helps ensure the application works fine.

**node\_modules** – The npm package installed is node\_modules. You can open the folder and see the packages available.

**src** – This folder is where we will work on the project using Angular 4.

The Angular 4 app folder has the following file structure –

**.angular-cli.json** – It basically holds the project name, version of cli, etc.

**.editorconfig** – This is the config file for the editor.

**.gitignore** – A .gitignore file should be committed into the repository, in order to share the ignore rules with any other users that clone the repository.

**karma.conf.js** – This is used for unit testing via the protractor. All the information required for the project is provided in karma.conf.js file.

**package.json** – The package.json file tells which libraries will be installed into node\_modules when you run npm install.

## 3.4 INTRODUCTION TO MIDDLEWARE

Middleware handles the data that comes from the frontend application as JSON Objects and these objects need to be processed before storing it to the database

### 3.4.1 EXPRESS MIDDLEWARE

The middleware is built by using express framework and nodejs runtime. The sample introduction code snippet is given below.

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

**fig 3.3 Middleware Init Express**

This app starts a server and listens on port 3000 for connections. The app responds

with “Hello World!” for requests to the root URL (/) or route. For every other path, it will respond with a 404 Not Found.



The example above is actually a working server: Go ahead and click on the URL shown. You'll get a response, with real-time logs on the page, and any changes you make will be reflected in real time. This is powered by RunKit, which provides an interactive JavaScript playground connected to a complete Node environment that runs in your web browser. Below are instructions for running the same app on your local machine.

### 3.4.2 FLASK MIDDLEWARE

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Hello, World!"
```

**Fig 3.4 Middleware Init Flask**

The above code snippet will start a server at the localhost of 127.0.0.1 at the default flask port of 5000

### 3.5 INTRODUCTION TO MONGODB

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schema. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL).

MongoDB scales horizontally using sharding.[11] The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more replicas.). Alternatively, the shard key can be hashed to map to a shard – enabling an even data distribution.

MongoDB can run over multiple servers, balancing the load or duplicating data to keep the system up and running in case of hardware failure.

### 3.6 REST API

**Client-server** – By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.

**Stateless** – Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.

**Cacheable** – Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.

**Uniform interface** – By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

**Layered system** – The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.

### **3.7 INTRODUCTION TO ETHEREUM**

Ether is the cryptocurrency generated by the Ethereum platform as a reward to mining nodes for computations performed and is the only currency accepted in the payment of transaction fees.

Ethereum provides a decentralized virtual machine, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes. The virtual machine's instruction set, in contrast to others like Bitcoin Script, is Turing-complete. "Gas", an internal transaction pricing mechanism, is used to mitigate spam and allocate resources on the network.

## CHAPTER 4

# DOCKER BASED DECENTRALIZED LOGISTICS MANAGEMENT SYSTEM

## 4.1 FRONTEND OPERATION

In my proposed system, the Angular frontend handles the user inputs and does the post request to the server through REST API the servers are connected in the micro services architecture. The angular app has distinct and shared components for the respected functions. Registration, Login forms utilize the same components and use the services that are injected into the components as dependency injection. Since the components are used to generate view in the browser, the services are used for manipulating the data and does most of the functions of the app and also takes the work of interacting with the server

### (i) COMPONENTS USED IN THE APP:

- |              |                |            |
|--------------|----------------|------------|
| • account    | • offer        | • footer   |
| • admin-dash | • supp-profile | • main-nav |
| • cart       | • admin        | • profile  |
| • home       | • buyer        | • supplier |

### (ii) SERVICES USED IN THE APP:

- |                |                |
|----------------|----------------|
| • Auth Service | • HTTP Handler |
|----------------|----------------|

(iii) **AUTH GUARD** is used for protection of the routes that require authentication and require special permission

(iv) **MATERIAL MODULE** is used for theming the application

## 4.2 MIDDLEWARE OPERATION

The endpoints in the middleware API provides the secure data handling capability of the frontend application. The middleware application has various route endpoints, they handle the respective user types.

### 4.2.1 ENDPOINTS IN THE API

The main endpoints in the middle ware application are listed below

- admin
- product
- supplier
- user

Another flask API middleware has the distinct endpoint for the calculated results of the transportation model.

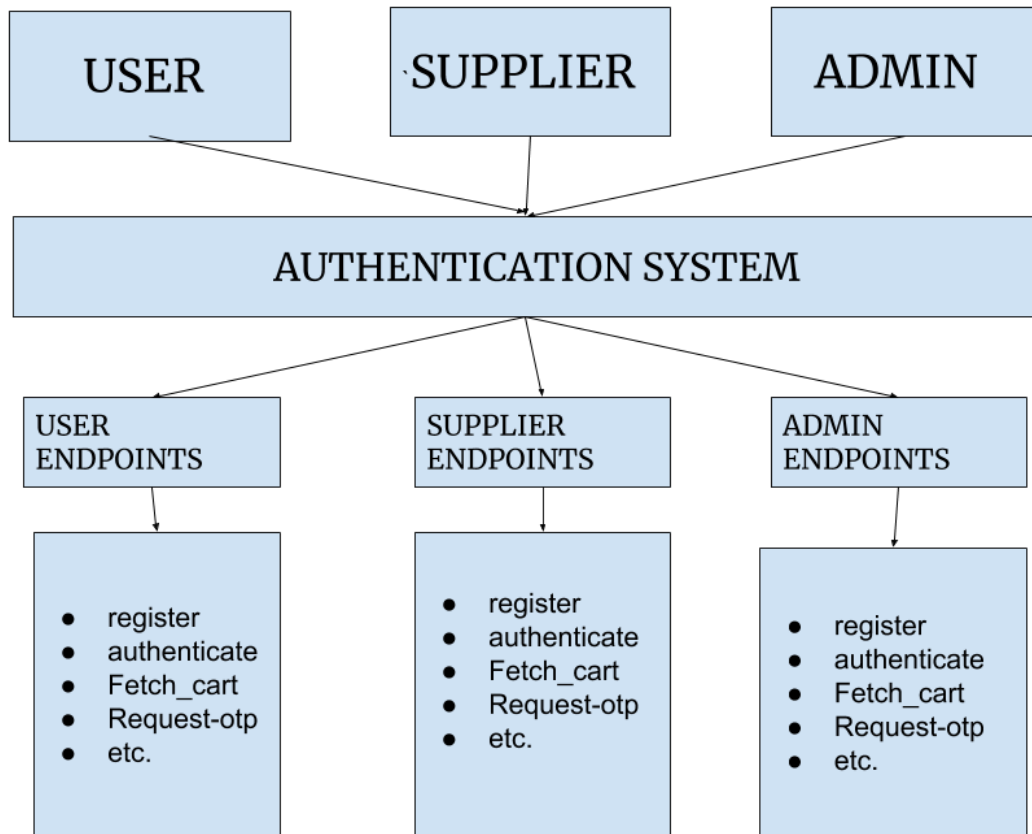
### 4.3 DATABASE

As mentioned above the MongoDB is the database used for the system. This database is configured in the middleware by using the mongoose package. In Mongoose, the schema is created as the models for the document oriented database, these schemas helps us to connect to the database of the application. The configuration for the database are available in the config directory.

### 4.4 ETHEREUM NETWORK

The main ethereum network has large number of users, for the development purposes, we can use the local ethereum network by using the ganache. It initiates local networks by creating maximum of 10 ether accounts and with 100 ether in wallet of the generated accounts. The ethereum network is connected to middleware by **web3**, this web3 client enables to connect to the local or main ethereum network. The local ethereum network initiated by the ganache is used for the monetary transactional puposes.It is noted that some of the 10 ether accounts is for suppliers and some accounts for actual users.

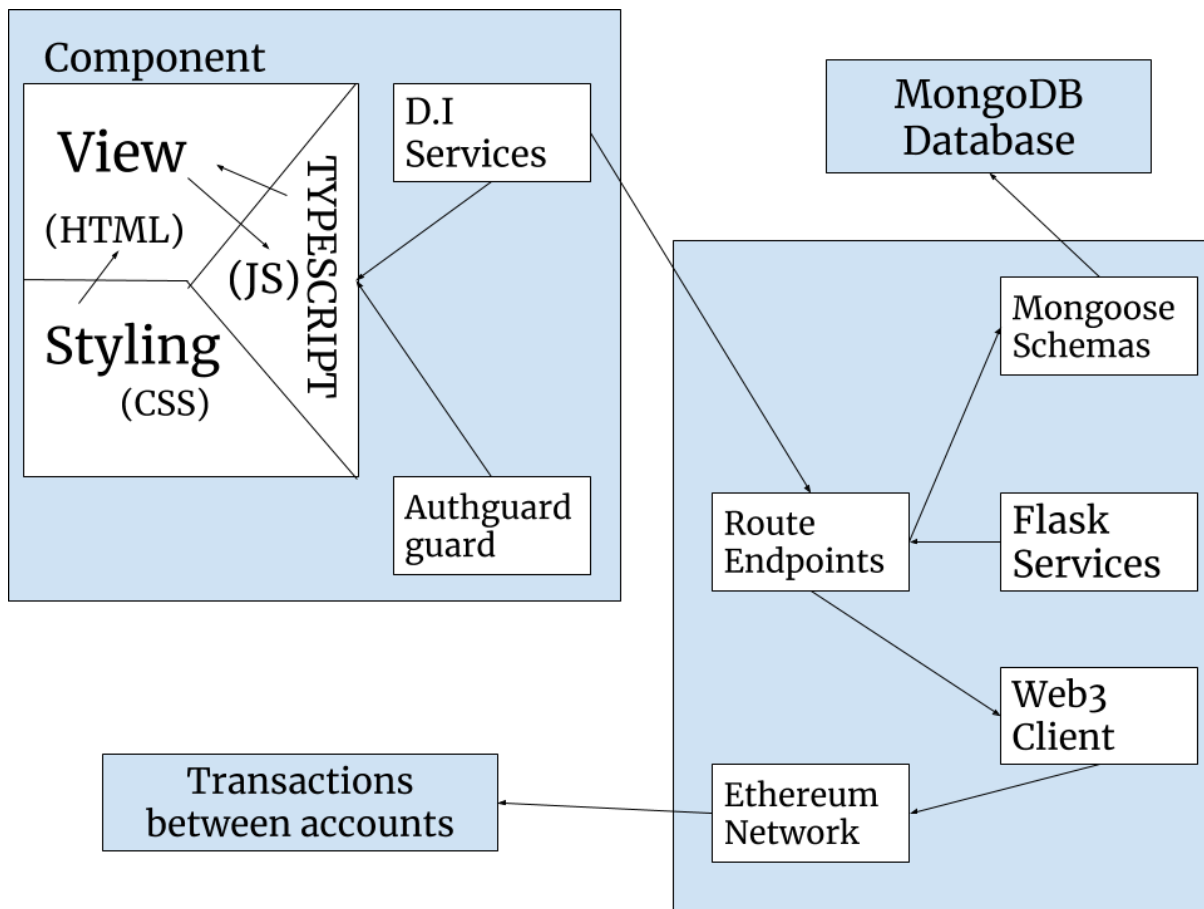
## 4.5 SYSTEM DESIGN



**Fig 4.1 System Design**

The respective users' login or registration data hit the server to the REST API in the json format as post request to the server this request is processed by the server and the API responds to the request in the json format to the frontend Angular application that is running on the browser.

## 4.6 ARCHTECTURAL DESIGN

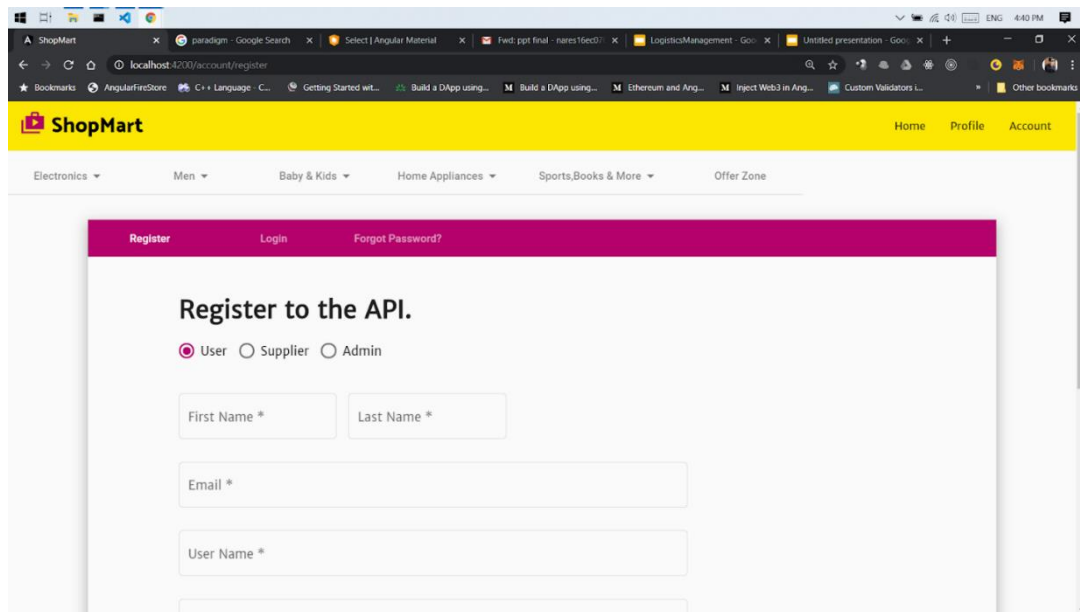


**Fig 4.2 Architectural Design**

The above diagram shows the data flow from the component of the application to the database and the ethereum network to store the data and to make the transactions respectively. This increases the limitations of the current application which is the core idea of the project.

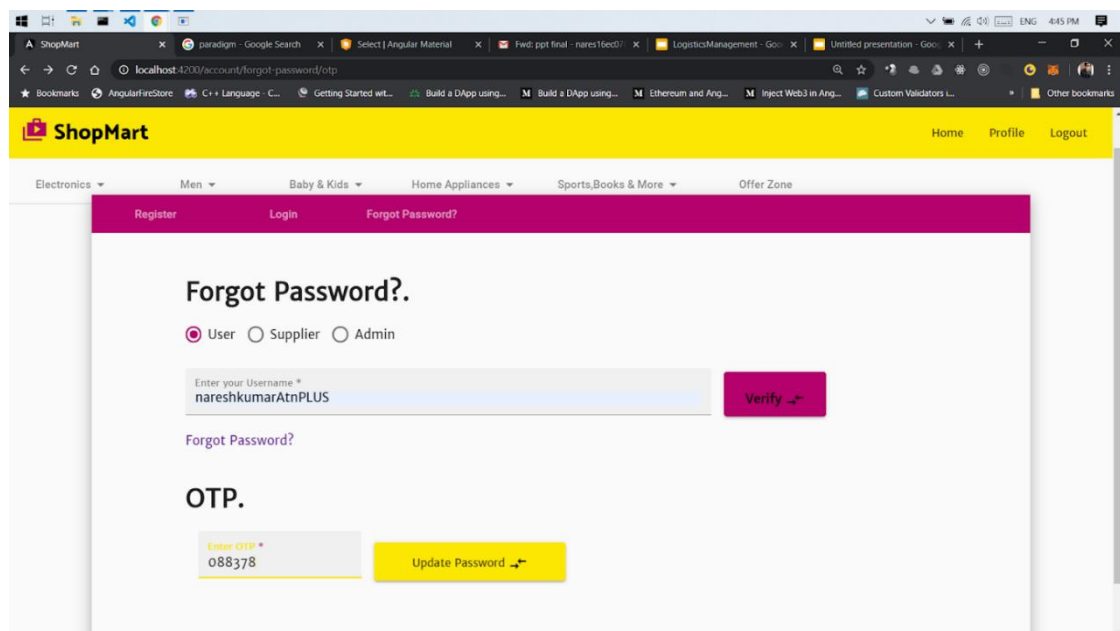
## CHAPTER 5

### RESULTS AND DISCUSSION



The screenshot shows the ShopMart website's registration page. The browser's address bar displays 'localhost:4200/account/register'. The page features a yellow header with the ShopMart logo and navigation links for Home, Profile, and Account. Below the header is a category menu with Electronics, Men, Baby & Kids, Home Appliances, Sports, Books & More, and Offer Zone. The main content area has a pink navigation bar with 'Register', 'Login', and 'Forgot Password?'. The 'Register' section is titled 'Register to the API.' and includes radio buttons for 'User' (selected), 'Supplier', and 'Admin'. There are input fields for 'First Name \*', 'Last Name \*', 'Email \*', and 'User Name \*'. A 'Register' button is partially visible at the bottom.

Fig 5.1 Registration Page



The screenshot shows the ShopMart website's forgot password page. The browser's address bar displays 'localhost:4200/account/forgot-password/otp'. The page features a yellow header with the ShopMart logo and navigation links for Home, Profile, and Logout. Below the header is a category menu with Electronics, Men, Baby & Kids, Home Appliances, Sports, Books & More, and Offer Zone. The main content area has a pink navigation bar with 'Register', 'Login', and 'Forgot Password?'. The 'Forgot Password?' section includes radio buttons for 'User' (selected), 'Supplier', and 'Admin'. There is an input field for 'Enter your Username \*' with the value 'nareshkumarAtnPLUS' and a 'Verify' button. Below this is a 'Forgot Password?' link. The 'OTP.' section has an input field for 'Enter OTP \*' with the value '088378' and an 'Update Password' button.

Fig 5.2 Forgot Password Page



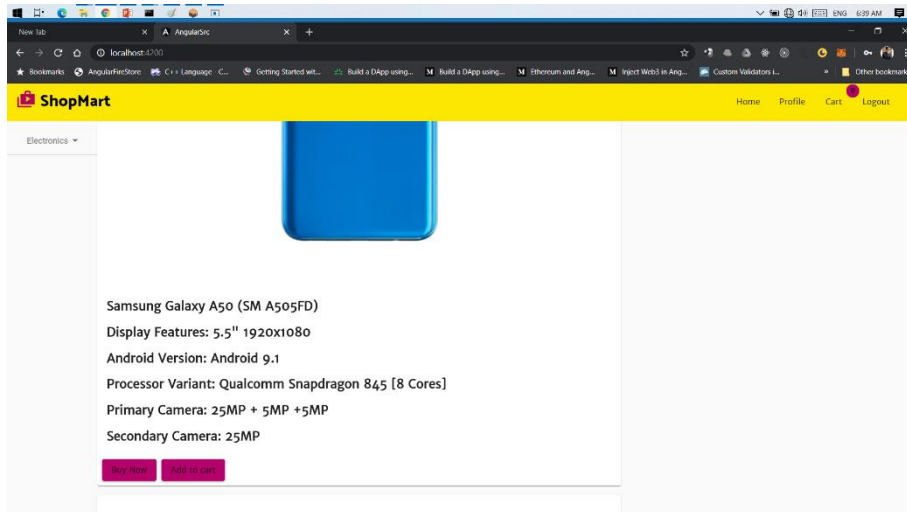


Fig 5.3 Product Page

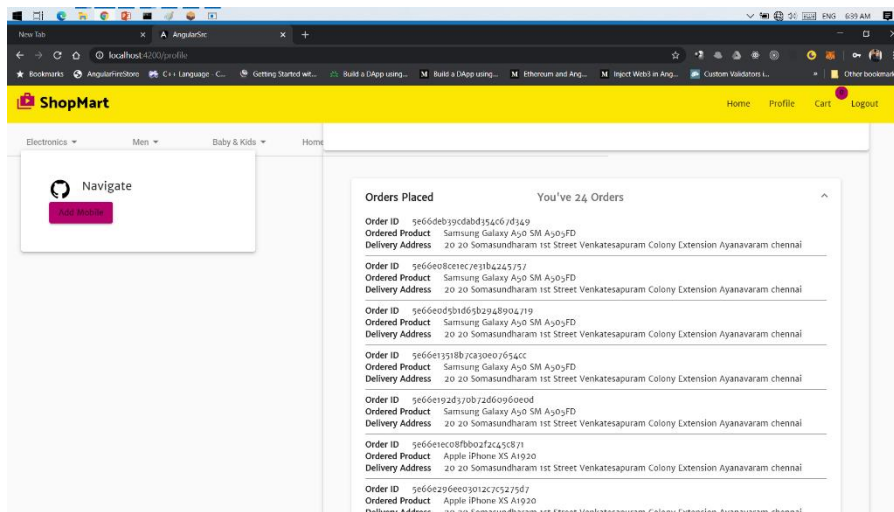
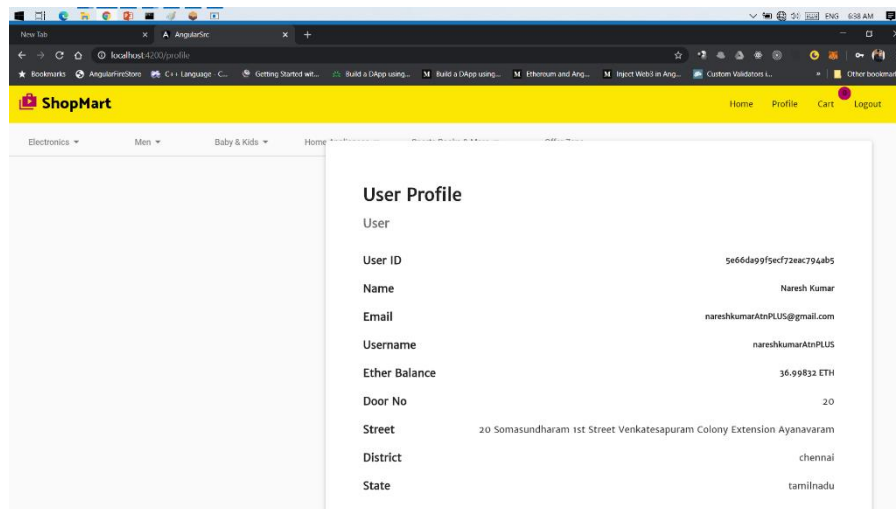
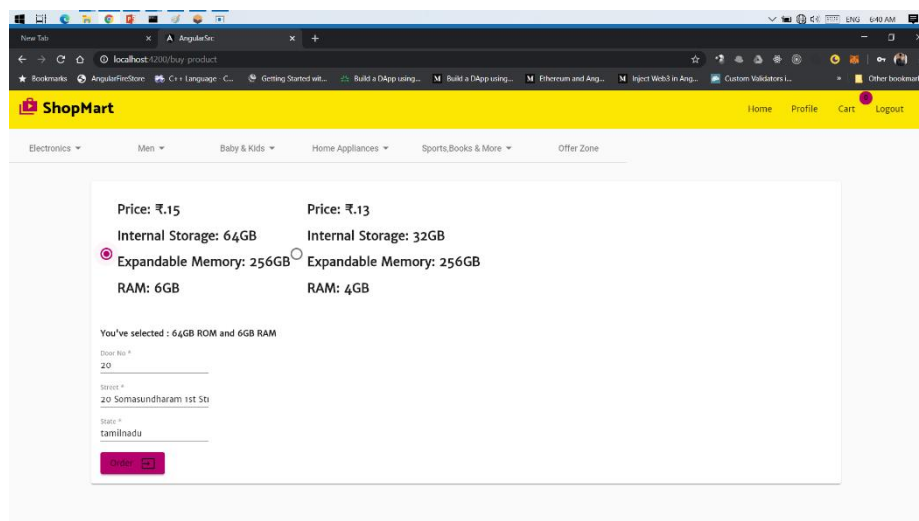


Fig 5.4 Seller Orders Page



**Fig 5.5 User Profile**



**Fig 5.6 Product Registration Page**

The above screenshots show the work flow of the application from registering to the decentralized e-commerce application to purchasing a mobile phone using ethers, a crypto currency. It shows the current user profile and the orders that the user have placed. It uses the OTP verification system for changing the password. This verification system sends the OTP through email to the user and the user needs to end the OTP while resetting the password.

## **CHAPTER 6**

### **CONCLUSION AND FUTURE WORK**

The conclusion were more vital for evaluating whether the system that designed enables the current disability and introduces a newer way for the usage of the application. From this project it is known that these are the present difficulites that meet to ease the application

- Developed a decentralized web application
- Optimized the costs in the logistics
- Improved the security in the monetary transactions
- Eliminated the involvement of third parties like banks etc.

These are the primary difficulties that are kept in clear in development of the product/project to ensure that these are completed to enable the current limitations.

## CHAPTER 7

### REFERENCES

1. Kannan Govindan, “Big data analytics and application for logistics and supply chain management”, Elsevier, International Journal of Logistics Management, Volume 114, June 2018, Pages 343-349 (2018).
2. Riccardo Giusti, “Synchronomodal logistics: An overview of critical success factors, enabling technologies, and open research issues”, Elsevier, International Journal of Logistics Management, Volume 129, September 2019, Pages 92-110 (2019).
3. Tsan-Ming Choi “Block chain-technology-supported platforms for diamond authentication and certification in luxury supply chain”, Elsevier, International Journal of Logistics Management, Volume 128, August 2019, Pages 17-29 (2019).
4. Tsan-Ming Choi, “Peer-to-peer collaborative consumption for fashion products in the sharing economy: Platform operation”, Elsevier, International Journal of Logistics Management, Volume 126, June 2019, Pages 49-65 (2019),
5. Chung-Shan Yang, “Maritime shipping digitalization: Block chain-based technology applications, future improvements, and intention to use”, Elsevier, International Journal of Logistics Management, Volume 131, November 2019, Pages 108-117,
6. angular, frontend, <https://angular.io/guide/architecture>
7. ethereum, network, <https://ethereum.org/developers/>
8. flask, <https://flask.palletsprojects.com/en/1.1.x/>
9. REST API [dratnerbewing.com/learnapidoc/docapis\\_what\\_is\\_a\\_rest\\_api.html](http://dratnerbewing.com/learnapidoc/docapis_what_is_a_rest_api.html)
10. JavaScript <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
11. nodejs, <https://nodejs.org/dist/latest-v12.x/docs/api/>
12. TypeScript <https://www.typescriptlang.org/docs/home.html>
13. express framework, <https://expressjs.com/en/4x/api.html>
14. firebase, <https://firebase.google.com/docs/web/setup>
15. docker, <https://docs.docker.com/>
16. Visual Studio Code, <https://code.visualstudio.com/docs>
17. Github, <https://guides.github.com/introduction/git-handbook/>
18. Git, <https://git-scm.com/doc>