

MINI PROJECT- II

(SESSION-2021-2022)

Build and deploy a website for ordering pizzas that satisfies the hunger of our customers in just few clicks.

Pizz360

Report



Institute of Engineering & Technology

Team

Members

Divyansh Garg

(181500222)

Kunal Singh

(181500341)

Supervised By

MR.ANAND GUPTA

TECHNICAL TRAINER

Department of Computer Engineering & Websites



Department of Computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

DECLARATION

I hereby declare that the work which is being presented in the Mini Project “**Pizz360**”, in partial fulfillment of the requirements for Mini project Lab is an authentic record of my own work carried under the supervision of **Mr. Anand Gupta, Technical Trainer.**

Divyansh Garg

Kunal Singh



Department of Computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

CERTIFICATE

This is to certify that the project entitled “Pizz360” carried out in Mini Project is legitimate work done by Divyansh Garg (181500222), Kunal Singh (181500341) and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Signature of Supervisor:

Name of Supervisor: Mr. Anand Gupta

Date: 25/11/2020





ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received.

Our heartiest thanks to Dr. (Prof). Anand Singh Jalal, Head of Dept., Department of CEA for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal.

We owe special debt of gratitude to Mr. Anand Gupta, Technical Trainer, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. He has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

DIVYANSH GARG

KUNAL SINGH

Abstract

You must have ordered your pizzas on phone for home delivery. The process seems easy to use but at times there is miscommunication. As there is no visual menu shown during a phone call, the employees have to repeat a lot of things again and again to the customers. It's a time consuming process which at times irritates customers. Also it takes a lot of time of the staff. It would be much more comfortable for the customers to have an online pizza ordering system. It would be hassle free for users as they can select the pizzas they want and make payment for it. Also it will reduce the purchasing time for customers. Let us look at another benefit of using this system. Suppose I go to a pizz360 and make order. Even after ordering pizzas from their outlet, I have to wait at least 15 minutes for my order to be ready. Wouldn't it be much more convenient if I ordered my pizzas before using a mobile app or an online system and then it will tell me the time by which I have to pick my order from their counter. It would be great for me as I don't need to wait for my pizza. I need to reach there only when my pizza is ready. In a nutshell, we can say that improved and efficient services are provided to the customers by the inclusion of internet in your business. As a business point of view it gives us an edge over our competitors.

Table of Contents

Declaration	2
Certificate	3
Acknowledgement	6
Abstract	7
Table Of Contents	8
1. Introduction	10
1.1 Motivation and Overview	10
1.2 Objective	11
2. Software Requirement Analysis	13
2.1 Define the Problem	13
2.2 Define the modules and their functionalities (SRS)	14
3. Software Design	21
3.1 Types of Software Design	21
3.2 Process	23
4. Software Testing	26
5. Requirements	27
5.1 Hardware	27
5.2 Software	27
6. Implementation and user interface	28
7. Conclusion	45

Pizza360

8. Future scope 45

9. References 45

Introduction

Pizz360 (Web-based Application) is a perfect mate for our Pizza Lovers, that stimulates the foodies (customers) to get their Pizzas Delivered at home in just one click. This will be a user-friendly application with proper menu tabs defined on homepage. It will have both Veg and Non-Veg Pizza's so that people can filter out based on their taste. Foodies can also customize their Pizza with their favorite toppings which will change the prices of these pizzas dynamically.

Once the user will place an order, an order id will be generated and provided to the user for making this whole process smooth. For this Application we will be using both Frontend and Backend Technologies and Database for storing the data.

Motivation

We all know that the online pizza ordering website is growing and there are lots of software available to provide these pizza ordering services but not that type of software which can provide at low price and best quality pizza directly to our consumers.

Overview

The main purpose of this project is to create an Onlinepizza ordering website that allows Consumers to order your desirable type of pizza and fast food.

This will be a user- friendly application with proper menu tabs defined on homepage. Foodies can place their order according to the menu.

Objective

This website offers our customers a superior product i.e., that will promote customer loyalty, at a low price, and provide customer service that is second to none. Its objective is to satisfy the customer's demand quality pizza that is delivered quickly with a smile.

Our goal is to deliver a website with a user interface (website) where customers can select various kinds of pizza and place their order. The order will be sent to the "kitchen" where the pizza will be made.

The focus is to create an "easy to use" website, which will allow a first time customer to complete their order with ease.

Hypothesis:

If these would be successfully done then the user or the common people can able to manage various type of modules available to manage pizza, Coupons, Payments. We can also generate reports for coupons, payments and online orders. Online order module manages online order operations. This application helps us to do all the functionalities more accurately and faster way. In future this website is helping in pizza ordering to maintain the stock and cash flow and many more functionalities, like

To store records

- Control order and services
- Control billing
- Control staff and their shifting
- Control multiple branches
- Helps manager to Control each part of the restaurant.

Operational Definitions:

Pizz360:

Our major objective will be to make this website as much user friendly as we can and satisfying the hunger of our customers in just few clicks.

How it works:

In this website, while accessing for the first time, customers' needs to register themselves by filling up basic registration details.

Once the registration is successful, customer need to login with a valid username and password for secure login.

The menu will be visible to the customer with the pizzas. All the ingredients will be shown with their prices.

After selecting a desired pizza, customers can view the details of pizza such as price, category and toppings.

Now payment option is shown to the customer. He can choose from the various online payment methods or cash on delivery option.

Customer can notify the admin about the system by writing a feedback message on contact us page.

About Pizz360

- Let customers to provide ratings and reviews for our pizzas.
- Simple, fast and convenient for ordering different kinds of pizzas.
- Menu with the actual pictures of the product thereby adding to the uniqueness of your online presence.
- Receive direct customer feedback and suggestions.
- Greater customer satisfaction!!!

Advantages:

- It overcomes all the problems of existing system.
- Pizza can be order in more convenient way.
- Payment can be easily done using various online mode or cash on delivery (COD)
- It makes system very effective for ordering a pizza.
- Admin can view sales report which can be helpful for decision making.
- Easy add/update/delete process of pizza.

Disadvantages:

- It requires a reliable internet connection.
- System may provide inaccurate results if data not entered correctly

Software Requirement Analysis

PROBLEM STATEMENT

Our solution is to make an ordering system that separates ordering pizzas from ordering the side dishes (non-pizza products) in an intuitive way. The ordering system provides the user with three tabs: one for ordering pizzas, one for side orders, and one for delivery details. Customers can switch to any tab anytime.

Hence the system will decrease workload of the employees and benefit the Pizz360 due to the database / information system. The system will be able to guide a user through the website and make then complete their pizza order. When they are done with filling in all information regarding their order they can complete the order to send it to the Pizz360.

DEFINITIONS

ABOUT FRONT-END TECHNOLOGY:

The front-end stack is made up of many different languages and libraries. While these vary from application to application, there are only a few generic languages understood by all web browsers. These three main front-end coding languages are HTML, CSS and JavaScript.

Together, they create the underlying scaffolding that web browsers use to render the web pages that we interact with every day. All other libraries and front-end engineering are built upon these three main languages, which makes them must-have skills for any front-end developer.

In fact, you can think of a webpage like a house. The initial UX design is the blueprint. HTML is the basic structure of the house. The CSS is the paint, fixtures, and other aesthetic decisions that make the house look attractive. And finally, JavaScript is the inner workings of the house (lights, heating, and water) that we, the owner or renter, use and enjoy.

VISUAL STUDIO:

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js and C++. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface, but can be accessed via the command palette.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

WEB BROWSER:

A web browser (commonly referred to as a browser) is a software application for accessing information on the World Wide Web. Each individual web page, image, and video is identified by a distinct Uniform Resource Locator (URL), enabling browsers to retrieve these resources from a web server and display them on the user's device.

A web browser is not the same thing as a search engine, though the two are often confused. For a user, a search engine is just a website, such as google.com, that stores searchable data about other websites. But to connect to a website's server and display its web pages, a user needs to have a web browser installed on their device.

The most popular browsers are Chrome, Firefox, Safari, Internet Explorer, and Edge.

Technical Feasibility

The proposed system is developed using HTML, CSS and bootstrap as front-end tool and PHP and JS node as the back end. The proposed system needs a Personal Web Server to serve the requests submitted by the users. The Web browser is used to view the web page that is available within the Windows operating system itself. The proposed system will run under Win9x, NT, and win2000 environment. As Windows is very user friendly and GUI OS it is very easy to use.

All the required hardware and software are readily available in the market. Hence the system is technically feasible.

Operational Feasibility:

The proposed system is operationally feasible because of the following reasons.

- The customer is benefited more as most of his time is saved. The customer is serviced at his place of work.
- The purpose of this website serves the good and needy people.

Economical Feasibility

As the necessary hardware and software are available in the market at a low cost, the initial investment is the only cost incurred and does not need any further enhancements. Hence it is economically feasible. The system is feasible in all respects and hence it encourages taking up the system design. We have used different languages and technologies for preparing the project.

LANGUAGES USED:

1. HTML5:

HTML stands for Hyper Text Mark-up Language. It is used to design web pages using mark-up language. HTML is the combination of Hypertext and Mark-up language. Hypertext defines the link between the web pages. Mark-up language is used to define the text document within tag which defines the structure of web pages.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

HTML code ensures the proper formatting of text and images so that your Internet browser may display them as they are intended to look. Without HTML, a browser would not know how to display text as elements or load images or other elements. HTML also provides a basic structure of the page, upon which Cascading Style Sheets are overlaid to change its appearance. One could think of HTML as the bones (structure) of a web page, and CSS as its skin (appearance).

2. CSS3:

Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

3. Bootstrap 4:

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites. It solves many problems which we had once, one of which is the cross-browser compatibility issue. . In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark colored tables, page headings, more prominent pull quotes, and text with a highlight. Bootstrap also comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the

functionality of some existing interface elements, including for example an auto-complete function for input fields.

Getting Started with Bootstrap Basics

Bootstrap is available in two forms; as a precompiled version, and as a source code version. The source code version uses the Less CSS preprocessor, but if you are more into Sass, there is an official Sass port of Bootstrap also available. To make it easier to make use of CSS vendor prefixes, Bootstrap uses Autoprefixer.

The source code version comes with source code written in Less (or Sass), all the JavaScript, and accompanying documentation. This allows more ambitious designers and developers to change and customize, at their will, all the provided styles, and to build their own version of Bootstrap.

4. JavaScript:

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross platform. JavaScript is the most popular programming language in the world and that makes it a programmer's great choice. Once you learnt JavaScript, it helps you developing great frontend as well as back-end software using different JavaScript based frameworks like jQuery, Node.JS etc.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

5. JQUERY

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

ABOUT BACK-END TECHNOLOGY:

1. MONGODB:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, Mongo DB uses JSON-like documents with optional schemas. Mongo DB is developed by Mongo DB Inc.

MongoDB supports field, range query, and regular-expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Queries can also be configured to return a random sample of results of a given size.

MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica-set member may act in the role of primary or secondary replica at any time.

2. Node JS:

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside 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-side and client-side scripts.

Node.js brings event-driven programming to web servers, enabling development of fast web servers in JavaScript. Developers can create scalable servers without using threading, by using a simplified model of event-driven programming that uses callbacks to signal the completion of a task. Node.js connects the ease of a scripting language (JavaScript) with the power of UNIX network programming.

3. Laravel :

Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things.

Laravel is a web application framework with expressive, elegant syntax. A web framework provides a structure and starting point for creating your application, allowing you to focus on creating something amazing while we sweat the details.

Laravel strives to provide an amazing developer experience, while providing powerful features such as thorough dependency injection, an expressive database abstraction layer, queues and scheduled jobs, unit and integration testing, and more.

Whether you are new to PHP or web frameworks or have years of experience, Laravel is a framework that can grow with you. We'll help you take your first steps as a web developer or give you a boost as you take your expertise to the next level. We can't wait to see what you build.

SOFTWARE DESIGN

Preliminary Design: Preliminary design is basically concerned with deriving an overall picture of the system. Deriving entire system into modules and sub-modules while keeping Cohesion and Coupling factors in mind. Tools, which assist in preliminary design process, are Data Flow Diagrams.

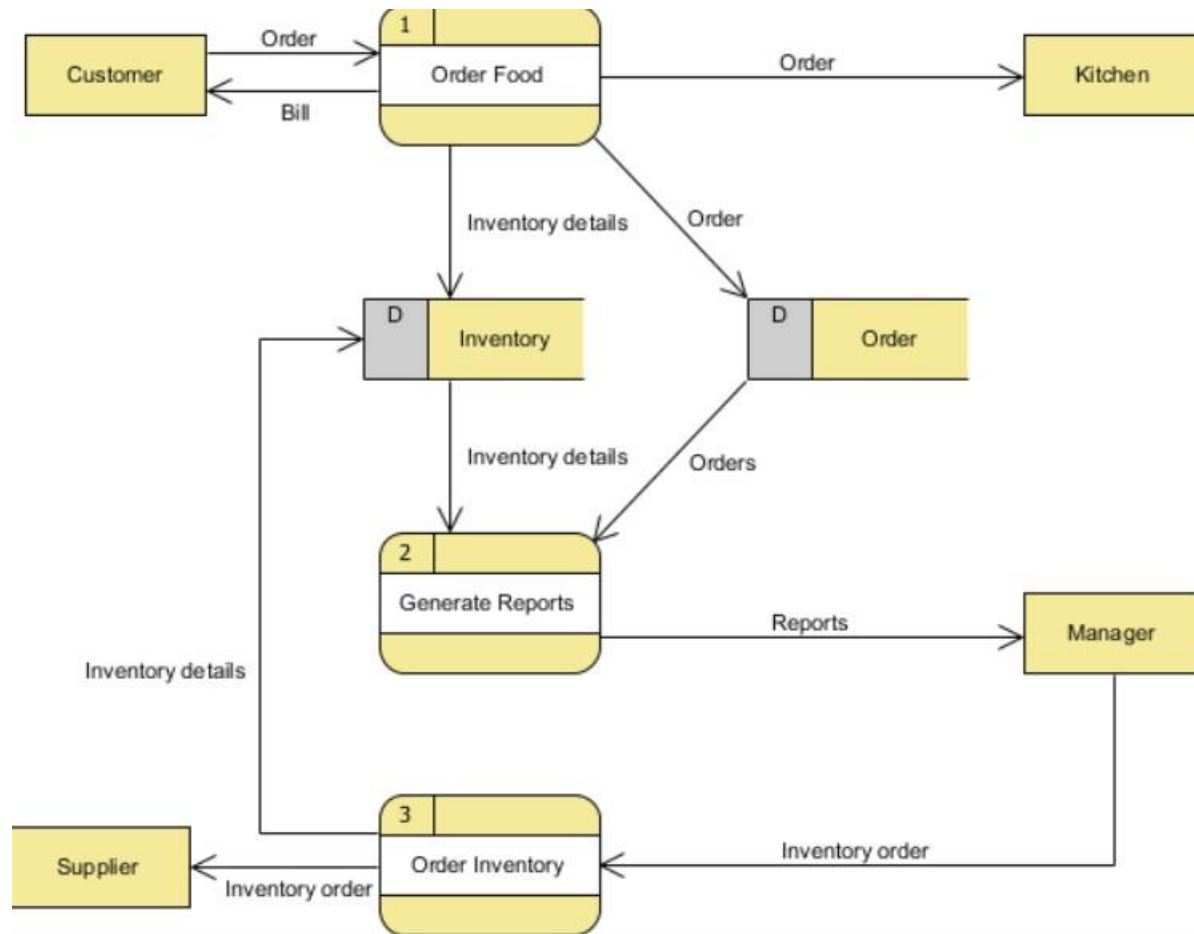
Code design: The purpose of code is to facilitate the identification and retrieval for items of information. A code is an ordered collection of symbols designed to provide unique identification of an entity or attribute. To achieve unique identification there must be only one place where the identified entity or the attribute can be entered in the code; conversely there must be a place in the code for everything that is to be identified. This mutually exclusive feature must be built into any coding system. The codes for this system are designed with two features in mind. Optimum human oriented use and machine efficiency. Length of the code range from length of one to length of five characteristics:

- The code structure is unique; ensuring that only one value of the code with a single meaning may be correctly applied to a given entity or attributes.
- The code structure is expansible allowing for growth of its set of entities and attributes.
- The code is concise and brief for recording, communication, transmission and storage efficiencies.
- They have a uniform size and format.
- The codes are simple so that the user can easily understand it.
- The codes are also versatile i.e., it is easy to modify to reflect necessary changes in condition, characteristic and relationships of the encoded entities.
- The codes are also easily storable for producing reports in a predetermined order of format.
- The codes are also stable and do not require being frequently updated thereby promoting user efficiency.
- The codes are also meaningful.
- They are also operable i.e., they are adequate for present and anticipate data processing both for machine and human use.

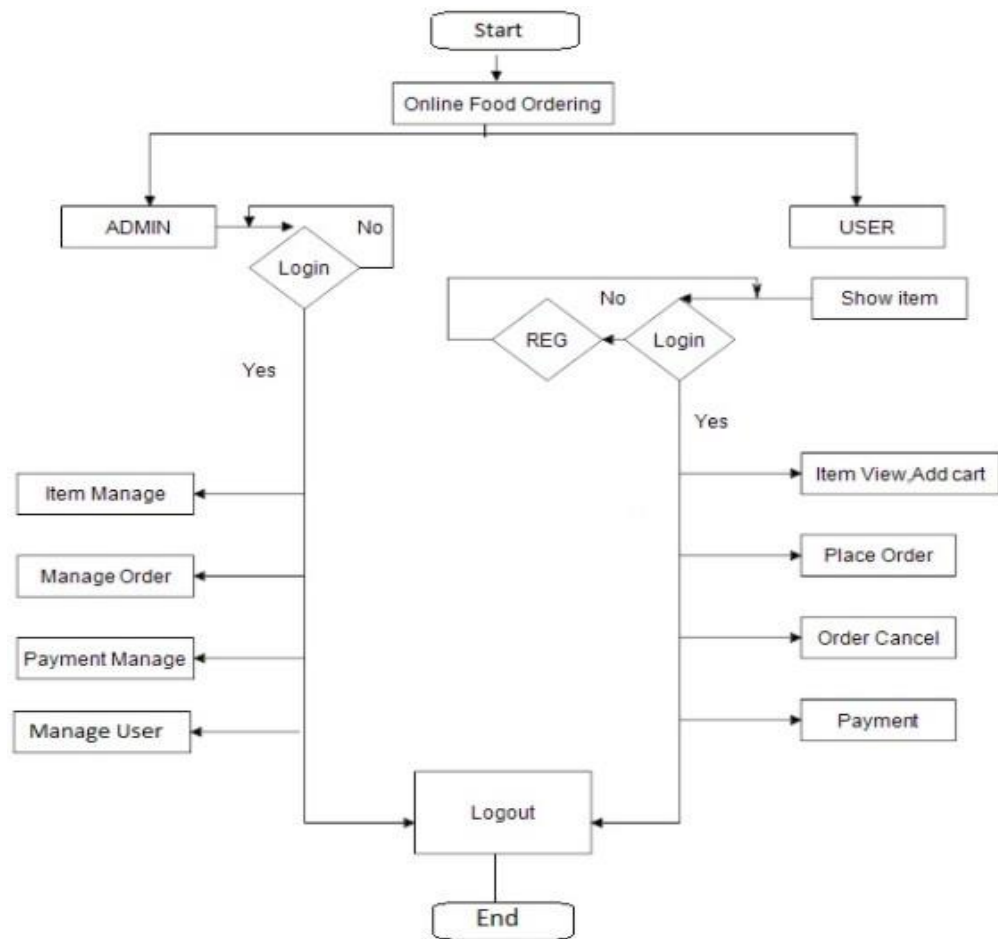
PROCESS



DataFlow Diagram



FlowChart



SOFTWARE TESTING

Testing is a process of executing a program with the intent of finding an error. Testing is a crucial element of software quality assurance and presents ultimate review of specification, design and coding. System Testing is an important phase. Testing represents an interesting anomaly for the software. A good test case is one that has a high probability of finding an as undiscovered error.

The test approach is divided into three main phases:

Module testing, integration tests and system testing.

In addition, the system testing includes two sub-phases: functional and usability testing. These planned tests are explained briefly below.

(a) Module testing will perform during coding by using debug messages to check that the written code produces wanted results. An important requirement is that the code will compile with zero bugs.

(b) Integration testing will perform after finish module testing in order to validate if each module can work fine with each other. Integration Test proves that system works as integrated unit when all the fixes are complete.

(c) System testing includes two phases: functional testing and usability testing. These will perform after the product reaches its final version. During functional test phase, the tester will test if the product meets the game requirements. The tester tests the requirements using the use cases listed below in Test Cases section. The usability test will perform to understand how easy it is to use this restaurant website. Any person out of the team members will perform this test by selecting the category and placing the order via call.

REQUIREMENTS:

Following are the hardware and the software requirements for our project:

a) Hardware:

Laptop/Desktop

1.8 GHz or faster processor. Quad-core or better recommended

4 GB of RAM

Hard disk space: Minimum of 800MB up to 210GB of available space

Video card that supports a minimum display resolution of 720p (1280 by 720)

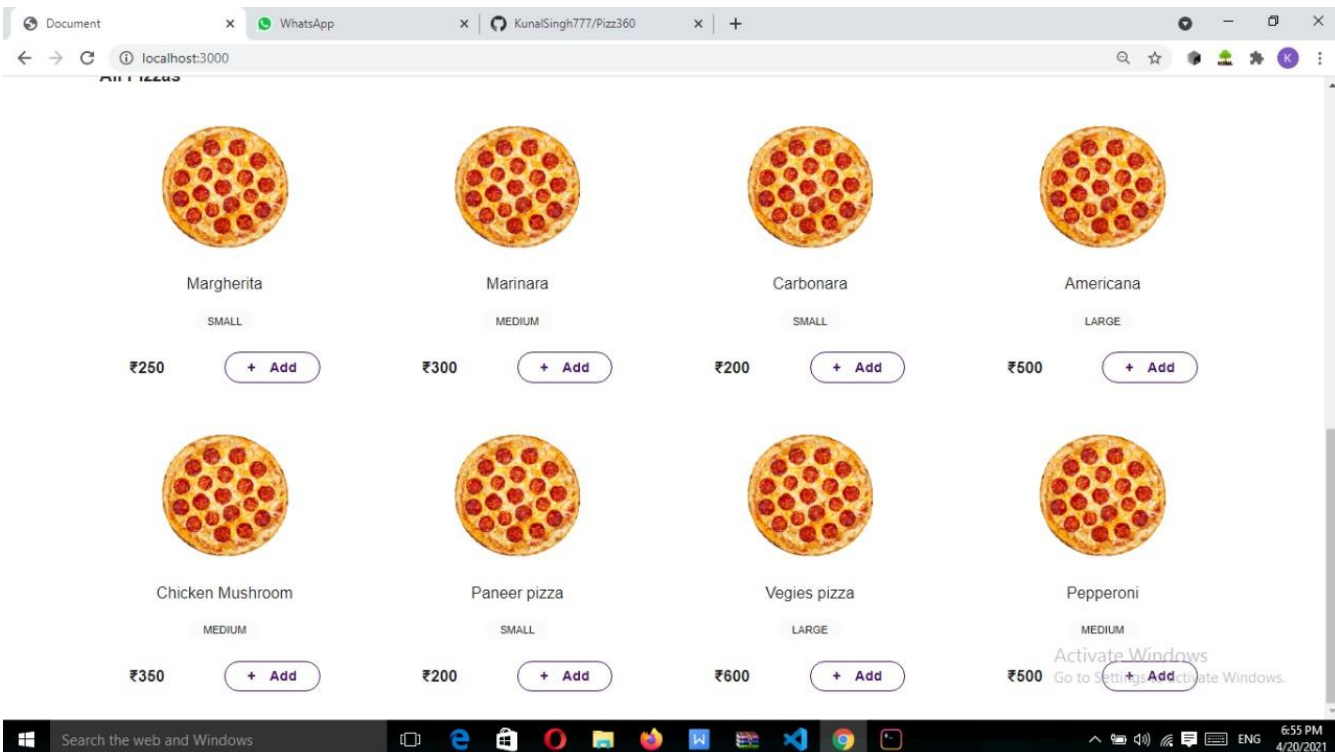
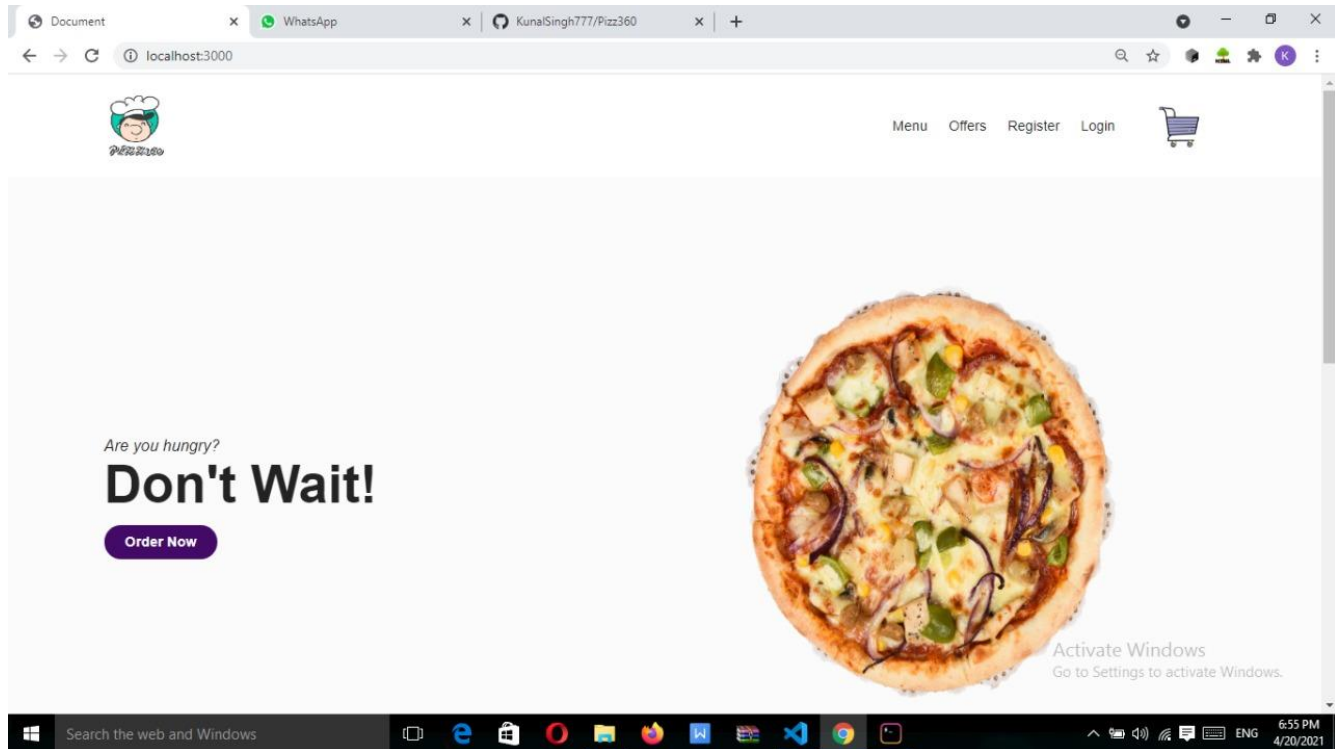
b) Software:

- Windows 8.1 and above
- Visual Studio
- HTML
- Web Browser
- Bootstrap
- MSSQL
- CSS
- NodeJS
- ReactJs
- MongoDB
- jQuery
- JavaScript

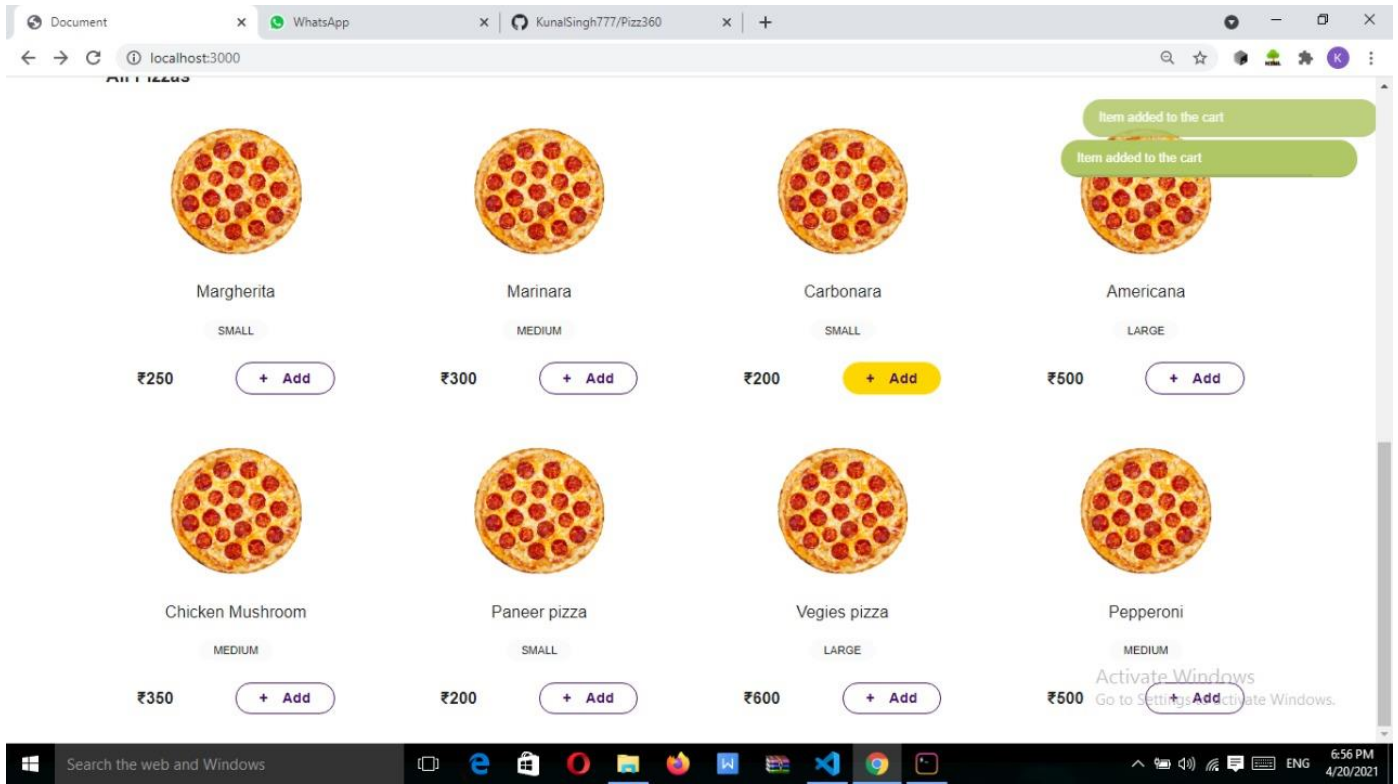
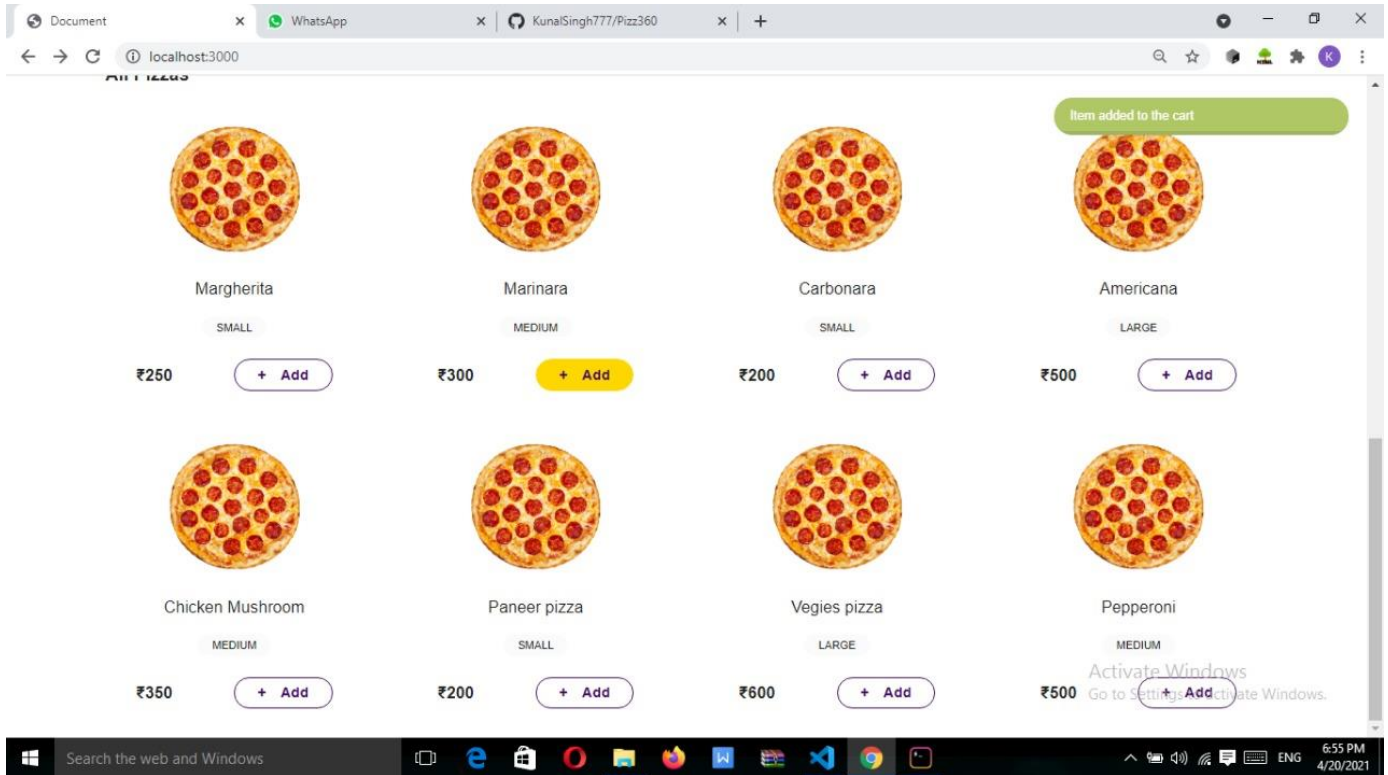
Pizza360

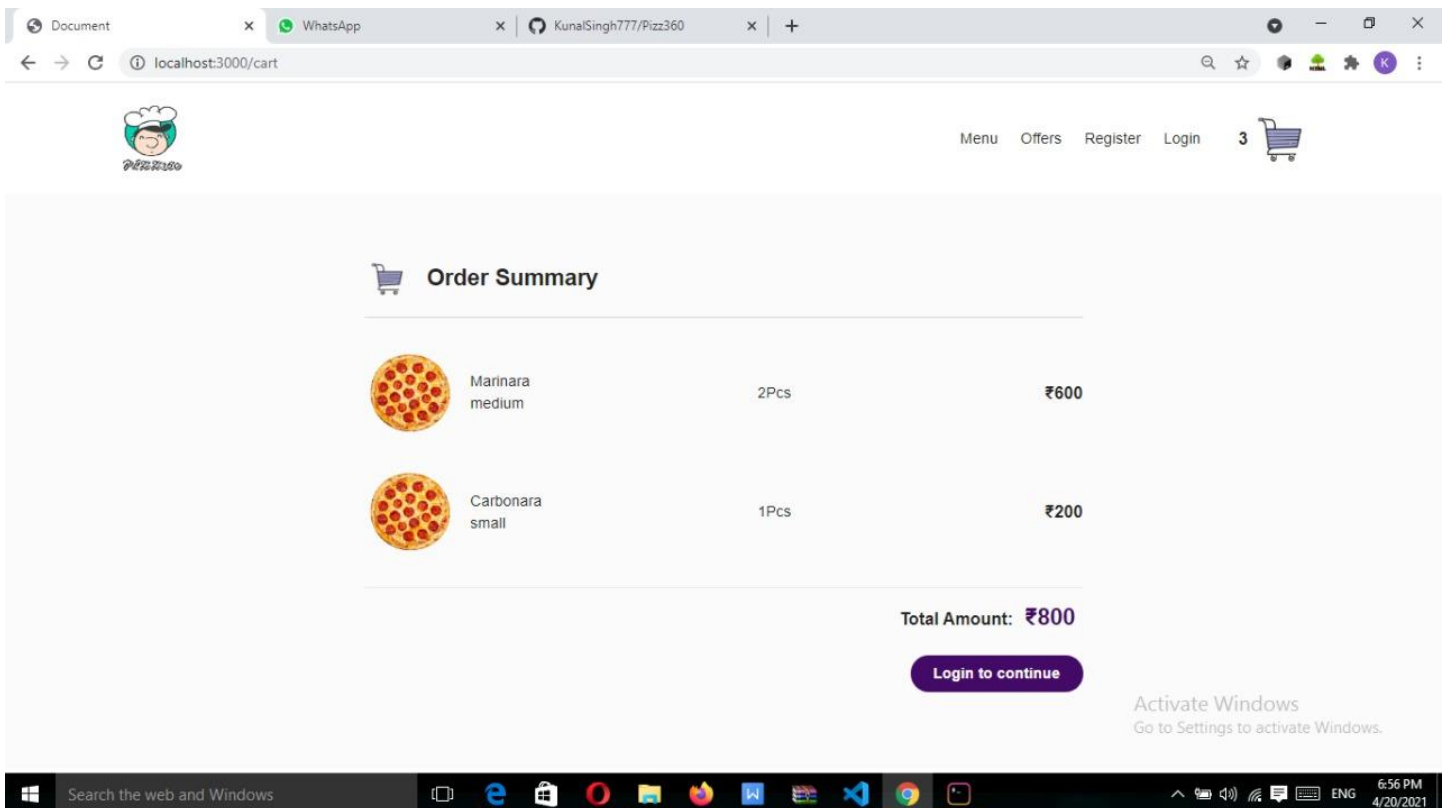
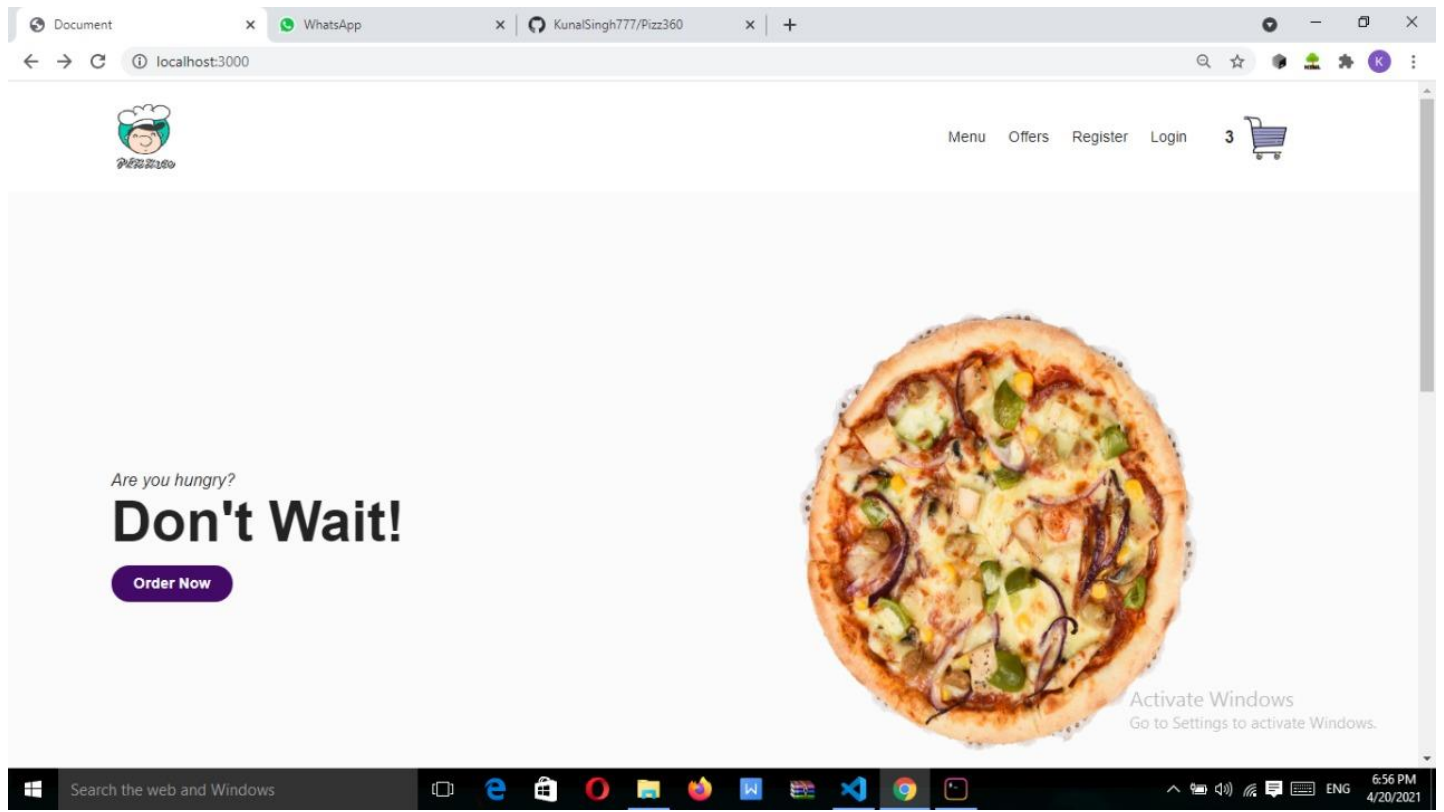
User Interface

Screenshots



Pizza360







Pizza360

Document x WhatsApp x KunaSingh777/Pizz360 x +

localhost:3000/login

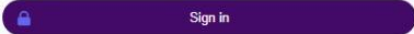
 Menu Offers Register Login 3 

Sign in to your account

Email address

Password

☐ Remember me [Don't have account?](#)





Activate Windows
Go to Settings to activate Windows.

Search the web and Windows

6:56 PM 4/20/2021

Document x WhatsApp x KunaSingh777/Pizz360 x +

localhost:3000/customer/orders

 Menu Offers Logout 3 

All Orders

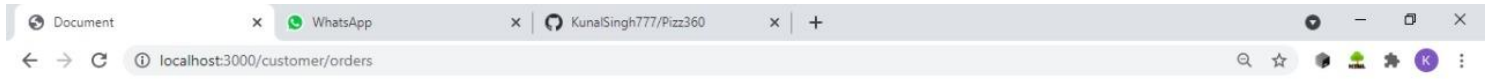
Orders	Phone	Address	Time
6065e5241b16a4277cbf8484	45665456	test address2423	08:52:PM
6065e4ed1b16a4277cbf8483	6784356366	test address242	08:51:PM
6065e32329cd9408502ec53e	34568988	test address24	08:43:PM
6065dff399c5650f487ff866	87654321	test address2	08:30:PM
60657bf521f7140c6cab3f07	9264955142	test address	01:23:PM

Activate Windows
Go to Settings to activate Windows.

Search the web and Windows

6:56 PM 4/20/2021

Pizza360

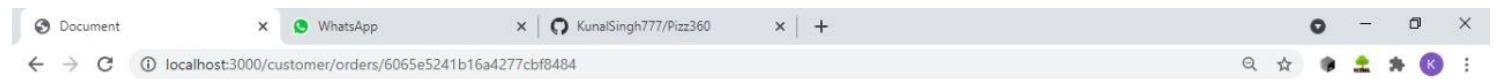


Menu Offers Logout 3

All Orders

Orders	Phone	Address	Time
6065e5241b16a4277cbf8484	45665456	test address2423	08:52:PM
6065e4ed1b16a4277cbf8483	6784356366	test address242	08:51:PM
6065e32329cd9408502ec53e	34568988	test address24	08:43:PM
6065dff399c5650f487ff866	87654321	test address2	08:30:PM
60657bf521f7140c6cab3f07	9264955142	test address	01:23:PM






Activate Windows
Go to Settings to activate Windows.



Menu Offers Logout 3

Track delivery status

6065e5241b16a4277cbf8484

-  Order Placed
-  Order confirmation
-  Preparation
-  Out for delivery
-  Complete

01:38 AM

Activate Windows
Go to Settings to activate Windows.



Document x WhatsApp x KunalSingh777/Pizz360 x +

localhost:3000/register

Menu Offers Register Login 3

Sign up

Name

Email address

Password

☐ Remember me

Register

Activate Windows
Go to Settings to activate Windows.

Search the web and Windows

6:57 PM
4/20/2021

Document x WhatsApp x KunalSingh777/Pizz360 x +

localhost:3000/cart

Order Summary

	Marinara medium	2Pcs	₹600
	Carbonara small	1Pcs	₹200

Total Amount: ₹800

Phone Number

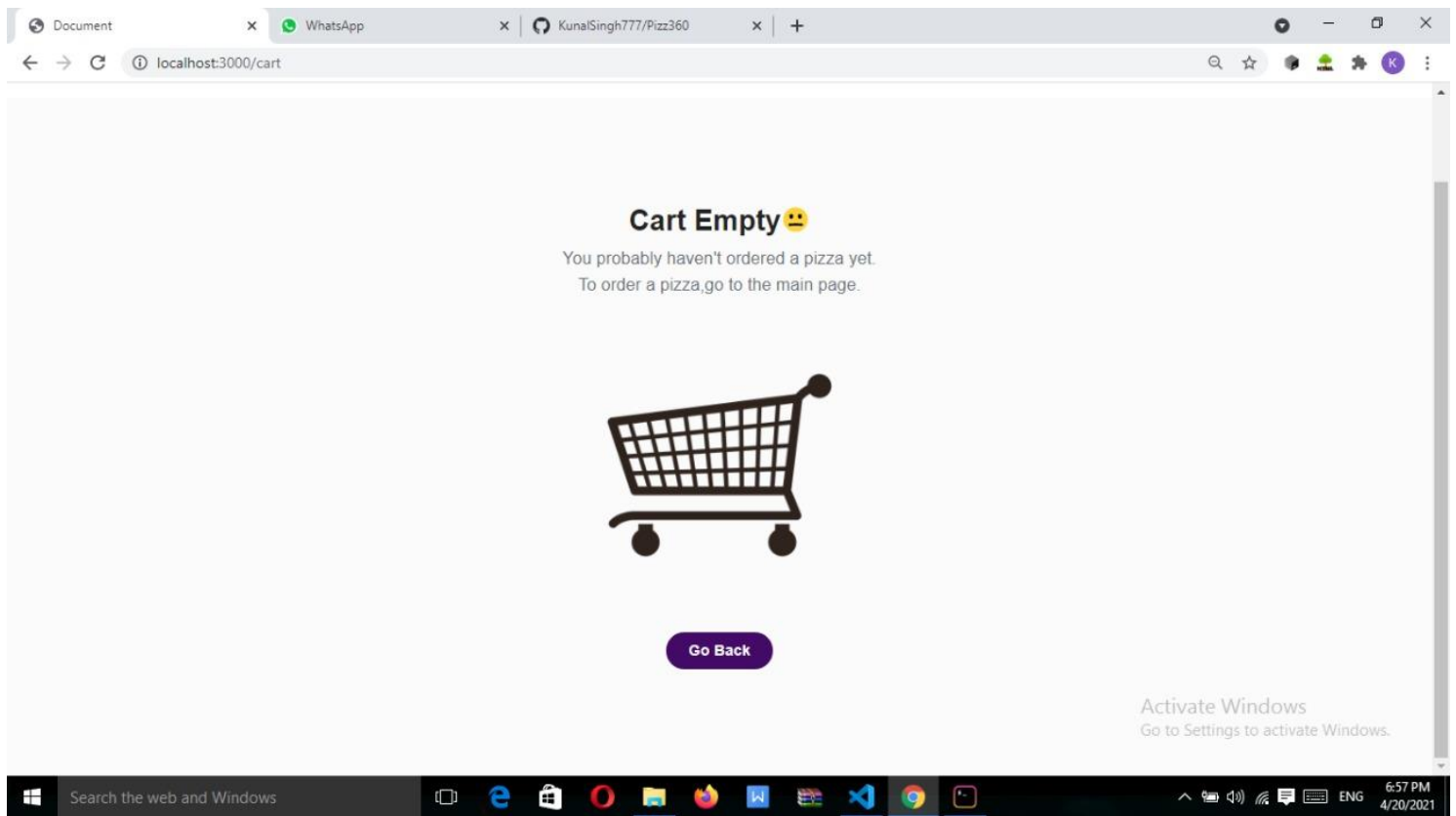
Address

Order Now

Activate Windows
Go to Settings to activate Windows.

Search the web and Windows

6:57 PM
4/20/2021



Document x WhatsApp x KunalSingh777/Pizz360 x +

localhost:3000/admin/orders

All orders

Orders	Customer	Address	status	Placed at	
607ed6c3ce6c3905608f2aea Marinara - 2 pcs Carbonara - 1 pcs	kunalsingh77	test address245	Placed	06:57 PM	Not paid
6065e5241b16a4277cbf8484 Marinara - 1 pcs Carbonara - 1 pcs Americana - 1 pcs	kunalsingh77	test address2423	Confirmed	08:52 PM	Not paid
6065e4ed1b16a4277cbf8483 Marinara - 1 pcs Paneer pizza - 1 pcs	kunalsingh77	test address242	Placed	08:51 PM	Not paid
6065e32329cd9408502ec53e Carbonara - 2 pcs Marinara - 3 pcs Margherita - 1 pcs Paneer pizza - 1 pcs Pepperoni - 1 pcs Americana - 3 pcs	kunalsingh77	test address24	Placed	08:43 PM	Not paid
6065dff399c5650f487f866 Carbonara - 1 pcs Marinara - 2 pcs Margherita - 1 pcs Paneer pizza - 1 pcs	kunalsingh77	test address2	Placed	08:30 PM	Not paid

Activate Windows
Go to Settings to activate Windows.

Pizza360

Document x WhatsApp x KunalSingh777/Pizz360 x +

localhost:3000/admin/orders

All orders

Orders	Customer	Address	status	Placed at	
607ed6c3ce6c3905608f2aea Marinara - 2 pcs Carbonara - 1 pcs	kunalsingh77	test address245	Placed	06:57 PM	Not paid
6065e5241b16a4277cbf8484 Marinara - 1 pcs Carbonara - 1 pcs Americana - 1 pcs	kunalsingh77	test address2423	Confirmed	08:52 PM	Not paid
6065e4ed1b16a4277cbf8483 Marinara - 1 pcs Paneer pizza - 1 pcs	kunalsingh77	test address242	Prepared	08:51 PM	Not paid
6065e32329cd9408502ec53e Carbonara - 2 pcs Marinara - 3 pcs Margherita - 1 pcs Paneer pizza - 1 pcs Pepperoni - 1 pcs Americana - 3 pcs	kunalsingh77	test address24	Placed	08:43 PM	Not paid
6065dff399c5650f487ff866 Carbonara - 1 pcs Marinara - 2 pcs Margherita - 1 pcs Paneer pizza - 1 pcs	kunalsingh77	test address2	Placed	08:30 PM	Not paid





Activate Windows
Go to Settings to activate Windows.

Search the web and Windows

6:58 PM
4/20/2021

Document x WhatsApp x KunalSingh777/Pizz360 x +

localhost:3000/cart

	Marinara medium	1Pcs	₹300
	Paneer pizza small	1Pcs	₹200
	Vegies pizza large	1Pcs	₹600
	Pepperoni medium	1Pcs	₹500

Total Amount: ₹1600

Phone Number

Address

Activate Windows
Go to Settings to activate Windows.

Search the web and Windows

6:59 PM
4/20/2021

Pizza360

The screenshot shows the MongoDB Compass interface. On the left, the 'Local' sidebar lists the database 'pizz360' and its collections: 'menus', 'orders', 'sessions', and 'users'. The main panel displays the 'Collections' tab for 'pizz360', showing a table of collections with their document counts, average document sizes, total document sizes, number of indexes, and total index sizes.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
menus	8	96.0 B	768.0 B	1	32.0 KB	
orders	6	788.7 B	4.6 KB	1	36.0 KB	
sessions	1	875.0 B	875.0 B	2	60.0 KB	
users	4	213.3 B	853.0 B	2	64.0 KB	

The screenshot shows the MongoDB Compass interface with the 'pizz360.menus' collection selected. The 'Documents' tab is active, displaying a list of documents. The top summary shows 8 documents, a total size of 768B, an average size of 96B, 1 index, and a total index size of 32.0KB. The documents are displayed in a JSON format, showing details like _id, name, image, price, and size for various pizza items.

Summary: DOCUMENTS 8, TOTAL SIZE 768B, AVG. SIZE 96B, INDEXES 1, TOTAL SIZE 32.0KB, AVG. SIZE 32.0KB

Documents:

- `{ "_id": "Seee651f739f8c674fd736ee", "name": "Margherita", "image": "pizza.png", "price": "250", "size": "small" }`
- `{ "_id": "Seee6671a27a66807cf2bea3", "name": "Marinara", "image": "pizza.png", "price": "300", "size": "medium" }`
- `{ "_id": "Seee6692a27a66807cf2bea4", "name": "Carbonara", "image": "pizza.png", "price": "200", "size": "small" }`
- `{ "_id": "Seee66a5a27a66807cf2bea5", "name": "Americana", "image": "pizza.png", "price": "300", "size": "medium" }`

Pizza360

The screenshot shows the MongoDB Compass interface for the `localhost:27017/pizz360.orders` collection. The left sidebar displays the database structure with `pizz360` expanded, showing collections like `orders`, `sessions`, and `users`. The main panel shows the `pizz360.orders` collection with 6 documents, a total size of 4.6KB, and 1 index. The document list displays two documents with details such as `_id`, `paymentType`, `status`, `customerId`, `items`, `phone`, `address`, `createdAt`, and `updatedAt`.

Local

9 DBS 14 COLLECTIONS

HOST localhost:27017

CLUSTER Standalone

EDITION MongoDB 4.4.3 Community

Filter your data

27017studentdb7

admin

as_db

config

local

pizz360

menus

orders

sessions

users

social_platform

MongoSH Beta

pizz360.orders

Documents

DOCUMENTS 6 TOTAL SIZE 4.6KB AVG. SIZE 789B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER OPTIONS FIND RESET

ADD DATA VIEW

Displaying documents 1 - 6 of 6 REFRESH

Activate Windows Go to Settings to activate Windows.

The screenshot shows the MongoDB Compass interface for the `localhost:27017/pizz360.sessions` collection. The left sidebar displays the database structure with `pizz360` expanded, showing collections like `orders`, `sessions`, and `users`. The main panel shows the `pizz360.sessions` collection with 1 document, a total size of 675B, and 2 indexes. The document list displays one document with details such as `_id`, `expires`, and `session`.

Local

9 DBS 14 COLLECTIONS

HOST localhost:27017

CLUSTER Standalone

EDITION MongoDB 4.4.3 Community

Filter your data

27017studentdb7

admin

as_db

config

local

pizz360

menus

orders

sessions

users

social_platform

MongoSH Beta

pizz360.sessions

Documents

DOCUMENTS 1 TOTAL SIZE 675B AVG. SIZE 675B INDEXES 2 TOTAL SIZE 60.0KB AVG. SIZE 30.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER OPTIONS FIND RESET

ADD DATA VIEW

Displaying documents 1 - 1 of 1 REFRESH

Activate Windows Go to Settings to activate Windows.

Pizza360

The screenshot shows the MongoDB Compass interface. On the left, the 'Local' sidebar lists databases and collections. The 'pizz360' database is expanded, showing collections: menus, orders, sessions, and users. The 'users' collection is selected. The main panel displays the 'pizz360.users' collection with 4 documents. The documents are:

- `{ "_id": ObjectId("606032d7690c83026c54a688"), "role": "customer", "name": "kunalSingh77", "email": "dd@gmail.com", "password": "$2b$10$CHQnWQVEEhpOUcT5oZCeSbnbavF310PamV.EGdgbX/37Vi0vWFC", "createdAt": 2021-03-28T07:40:07.608+00:00, "updatedAt": 2021-03-28T07:40:07.608+00:00, "__v": 0 }`
- `{ "_id": ObjectId("6060990e2d56260c54fd00af"), "role": "customer", "name": "lucky", "email": "johnwickkunal@gmail.com", "password": "$2b$10$X0H/cZPtwhjk93qkeeIELeg4tq3C9ISX8kociduvatsnWsp5Zj3ua", "createdAt": 2021-03-28T14:56:14.916+00:00, "updatedAt": 2021-03-28T14:56:14.916+00:00, "__v": 0 }`
- `{ "_id": ObjectId("6065f80c01f0e8102c22d300"), "role": "admin", "name": "admin_kunalSingh77", "email": "admindd@gmail.com", "password": "$2b$10$CHQnWQVEEhpOUcT5oZCeSbnbavF310PamV.EGdgbX/37Vi0vWFC", "createdAt": 2021-03-28T07:40:07.608+00:00, "updatedAt": 2021-03-28T07:40:07.608+00:00, "__v": 0 }`



The interface also shows a summary of the collection: 4 documents, 653B total size, 213B avg size, 2 indexes, 64.0KB total index size, and 32.0KB avg index size. A watermark 'Activate Windows' is visible in the bottom right corner.

The screenshot shows a web browser with three tabs: 'Document', 'WhatsApp', and 'KunalSingh777/Pizz360'. The active tab is 'KunalSingh777/Pizz360', which displays the Pizza360 application. The URL bar shows 'localhost:3000/customer/orders'. The application has a header with a Pizza360 logo, a navigation menu (Menu, Offers, Logout), and a shopping cart icon with a count of 4. The main content area is titled 'All Orders' and contains a table with columns: Orders, Phone, Address, and Time. The table is currently empty, with the text 'No orders found' displayed below it. A watermark 'Activate Windows' is visible in the bottom right corner.

Pizza360

Document x WhatsApp x KunalSingh777/Pizz360 x +

localhost:3000/customer/orders

 Menu Offers Logout 

All Orders

i Order placed successfully

Orders	Phone	Address	Time
607ed837ce6c3905608f2aeb	9264955142	lucknow	07:03:PM



Activate Windows
Go to Settings to activate Windows.

Search the web and Windows






7:03 PM
4/20/2021

Document x WhatsApp x KunalSingh777/Pizz360 x +

localhost:3000/customer/orders/607ed837ce6c3905608f2aeb

 Menu Offers Logout 

Track delivery status 607ed837ce6c3905608f2aeb

-  Order Placed
-  Order confirmation
-  Preparation
-  Out for delivery
-  Complete

Activate Windows
Go to Settings to activate Windows.

Search the web and Windows

7:03 PM
4/20/2021

Implementation:

Server.js

```

require('dotenv').config()
const express=require('express');
const app=express()
const PORT=process.env.PORT || 3000
const ejs=require('ejs')
const path=require('path')
const expressLayout=require('express-ejs-layouts')
const mongoose=require('mongoose')
const session=require('express-session')
const flash=require('express-flash')
const MongoDBStore=require('connect-mongo')(session)
const passport=require('passport')

// Database Cpnnection
const url='mongodb://localhost/pizz360'
mongoose.connect(url,{useNewUrlParser:true,useCreateIndex:true,useUnifiedTopology:true,useFindAndModify:true});
const connection=mongoose.connection;
connection.once('open',()=>{
  console.log("Database connected..");
}).catch(err=>{
  console.log("Connection failed...");
});

//Session store

let mongoStore=new MongoDBStore({
  mongooseConnection:connection,
  collection:'sessions'
})

//Session config
app.use(session({
  secret:process.env.COOKIE_SECRET,
  resave:false,
  store:mongoStore,
  saveUninitialized:false,
  cookie:{maxAge:1000*60*60*24}
})

```



```

    )))

//passport config
const passportInit=require('./app/config/passport')
passportInit(passport)
app.use(passport.initialize())
app.use(passport.session())

app.use(flash())
app.use(express.urlencoded({ extended:false }))

app.use(express.json())

app.use((req,res,next)=>{
  res.locals.session=req.session
  res.locals.user=req.user
  next()
})
//Assets
app.use(express.static('public'));
//set templates engine
app.use(expressLayout)
app.set('views',path.join(__dirname,'/resources/views'))
app.set('view engine','ejs')

require('./routes/web')(app)

app.listen(PORT,()=>{
  console.log("Listen on port 3000");
})

```

app/config/passport.js

```

const LocalStrategy=require('passport-local').Strategy
const User=require('../models/user')
const bcrypt=require('bcrypt')
function init(passport){
  passport.use(new LocalStrategy({ usernameField:'email'},async (email,password,done)=>{
    const user=await User.findOne({ email:email })
    if(!user){
      return done(null,false,{ message:'No user with this email' })
    }
    bcrypt.compare(password,user.password).then(match=>{
      if(match){

```



```

        return done(null,user,{ message:'Logged in successfull'})
      }
      return done(null,false,{ message:'Wrong username or password'})
    }).catch(err=>{
      return done(null,false,{ message:'Something went wrong'})
    })
  })
}

passport.serializeUser((user,done)=>{
  done(null,user._id)
})
passport.deserializeUser((id,done)=>{
  User.findById(id,(err,user)=>{
    done(err,user)
  })
})
}

module.exports=init

```

app/http/controllers

```

const User=require('../models/user')
const bcrypt=require('bcrypt');
const passport = require('passport');
function authController(){
  const _getRedirectUrl = (req) => {
    return req.user.role === 'admin' ? '/admin/orders' : '/customer/orders'
  }

  return{
    login(req,res){
      res.render('auth/login');
    },
    postLogin(req,res,next){
      passport.authenticate('local',(err,user,info)=>{
        if(err){
          req.flash('error',info.message)
          return next(err)
        }
        if(!user){
          req.flash('error',info.message)
          return res.redirect('/login')
        }
        req.logIn(user,(err)=>{

```

```

        if(err){
            req.flash('error',info.message)
            return next(err)
        }
        return res.redirect(_getRedirectUrl(req))
    })
})(req,res,next)
},
register(req,res){
    res.render('auth/register');
},
async postRegister(req,res){
    const { name,email,password}=req.body
    //validate request
    if(!name || !email || !password){
        req.flash('error','All fields are required')
        return res.redirect('/register')
    }
    //check if email exists

    User.exists({ email:email },(error,result)=>{
        if(result){
            req.flash('error','Email already exists')
            return res.redirect('/register')
        }
    })

    //Hash password
    const hashedPassword= await bcrypt.hash(password,10)
    //Create a user

    const user=new User({
        name:name,
        email:email,
        password:hashedPassword
    })
    user.save().then((user)=>{
        //login
        return res.redirect('/')
    }).catch(error=>{
        req.flash('error','Something went wrong')
        return res.redirect('/register')
    })

},
logout(req,res){
    req.logout()
    return res.redirect('/login')
}

```

Pizza360

```
    }  
  }  
}
```

module.exports=authController

app/http/controllers/homeController.js

```
const Menu=require('../../models/menu')  
function homeController()  
{  
  return{  
    index(req,res)  
    {  
      Menu.find().then(function(pizzas){  
  
        return res.render('home',{pizzas:pizzas})  
      })  
    }  
  }  
}
```

module.exports=homeController

app/http/controllers/customers/cartController.js

```
function cartController(){  
  return{  
    index(req,res){  
      res.render('customers/cart');  
    },  
    update(req,res){  
  
      if(!req.session.cart){  
        req.session.cart={  
          items:{},  
          totalQty:0,  
          totalPrice:0  
        }  
      }  
      let cart=req.session.cart  
      if(!cart.items[req.body._id]){
```

```

        cart.items[req.body._id]={
            item:req.body,
            qty:1
        }
        cart.totalQty=cart.totalQty+1
        cart.totalPrice=cart.totalPrice+req.body.price
    }
    else{
        cart.items[req.body._id].qty=cart.items[req.body._id].qty+1;
        cart.totalQty=cart.totalQty+1
        cart.totalPrice=cart.totalPrice+req.body.price
    }
    return res.json({ totalQty:req.session.cart.totalQty});
}
}
}

module.exports=cartController

```

app/http/controllers/customers/orderController.js

```

const Order=require('../../models/order')
const moment=require('moment')
function orderController(){
    return{
        store(req,res){
            //validate request
            const{ phone,address}=req.body
            if(!phone || !address){
                req.flash('error','All fields are required')
                return res.redirect('/cart')
            }

            const order=new Order({
                customerId:req.user._id,
                items:req.session.cart.items,
                phone,
                address
            })

            order.save().then(result=>{
                req.flash('success','Order placed successfully')
                delete req.session.cart
                return res.redirect('/customer/orders')
            }).catch(err=>{

```

Pizza360

```
    req.flash('error','Something went wrong')
    return res.redirect('/cart')
  })
},
async index(req,res){
  const orders=await Order.find({customerId:req.user._id},null,{ sort:{'createdAt':-1}})
  res.header('Cache-Control', 'no-store')
  res.render('customers/orders',{ orders:orders,moment:moment })

},
async show(req,res){
  const order= await Order.findById(req.params.id)
  //Authorize User
  if(req.user._id.toString()===order.customerId.toString())
  {
    return res.render('customers/singleOrder',{ order })
  }
  return res.redirect('/')
}
}
}

module.exports=orderController
```

main/pizza-menu.json

```
[{
  "_id": {
    "$oid": "5eee651f739f8c674fd736ee"
  },
  "name": "Margherita",
  "image": "pizza.png",
  "price": "250",
  "size": "small"
},{
  "_id": {
    "$oid": "5eee6671a27a66807cf2bea3"
  },
  "name": "Marinara",
  "image": "pizza.png",
  "price": "300",
  "size": "medium"
},{
  "_id": {
```

Pizza360

```
"$oid": "5eee6692a27a66807cf2bea4"
},
"name": "Carbonara",
"image": "pizza.png",
"price": "200",
"size": "small"
},{
  "_id": {
    "$oid": "5eee66a5a27a66807cf2bea5"
  },
  "name": "Americana",
  "image": "pizza.png",
  "price": "500",
  "size": "large"
},{
  "_id": {
    "$oid": "5eee66c4a27a66807cf2bea6"
  },
  "name": "Chicken Mushroom",
  "image": "pizza.png",
  "price": "350",
  "size": "medium"
},{
  "_id": {
    "$oid": "5eee66cfa27a66807cf2bea7"
  },
  "name": "Paneer pizza",
  "image": "pizza.png",
  "price": "200",
  "size": "small"
},{
  "_id": {
    "$oid": "5eee66eea27a66807cf2bea8"
  },
  "name": "Vegies pizza",
  "image": "pizza.png",
  "price": "600",
  "size": "large"
},{
  "_id": {
    "$oid": "5eee6717a27a66807cf2bea9"
  },
  "name": "Pepperoni",
  "image": "pizza.png",
  "price": "500",
  "size": "medium"
}]
```

main/resources/scss/app.scss

```
@import './variables';
@import '~noty/src/noty.scss';
@import '~noty/src/themes/mint.scss';
```

```
body{
  font-family: 'Lato',sans-serif;
  -webkit-font-smoothing:antialiased;
  color:$dark;
}
```

```
section.hero{
  background: $secondary;
}
```

```
.size{
  background-color: $secondary;
}
```

```
.add-to-cart{
  border: 2px solid #440a67;
  color: #440a67;
  transition: $smooth;
  &:hover{
    background-color: gold;
    color: #440a67;
    border-color: gold;
  }
  &:focus{
    outline: none;
  }
}
```

```
.btn-primary{
  background-color:#440a67 ;
}
```

```
.btn-primary:hover{
  color:#440a67;
  background-color: gold;
  border-color: gold;
  transition: $smooth;
}
```

```
section.cart{
  background-color: $secondary;
  .amount{
```

Pizza360

```
    color: #440a67;
  }
}
.noty_theme__mint{
  border-radius: 50px!important;
  padding-left: 8px!important;
}
a.link{
  color:red;
}
section.status{
  background: $secondary;
  min-height: calc(100vh-86px);
}
.status-box{
  padding: 60px 0 0;
}
.status-box ul{
  margin-left: 84px;
}
.status-box ul li span{
  position: relative;
  padding-left: 20px;
}
.status-box ul li span:after{
  content: "";
  position: absolute;
  left: -10px;
  top: 50%;
  background-color: $dark;
  width: 10px;
  height: 10px;
  border-radius: 50%;
  transform: translateY(-50%);
  box-shadow: 0 0 1px 10px $secondary;
}
.status-box ul li span:before{

  font-family: "Line Awesome Free";
  font-size: 46px;
  font-weight: 600;
  position: absolute;
  left: -90px;
  top: 50%;
  transform: translateY(-50%);
  color: inherit;
}
.status-box ul li:nth-child(1) span:before{
```



```

    content: '\f46c';
}
.status-box ul li:nth-child(2) span:before{
    content: '\f560';
}
.status-box ul li:nth-child(3) span:before{
    content: '\f818';
}
.status-box ul li:nth-child(4) span:before{
    content: '\f0d1';
}
.status-box ul li:nth-child(5) span:before{
    content: '\f582';
}
.status-box ul li{
    position: relative;
}
.status-box ul li:before{
    content: "";
    position: absolute;
    top: 0;
    left: -6px;
    width: 2px;
    height: 100%;
    background: $dark;
    margin-top: 10px;
}
.status-box ul li:nth-child(5) span:after{
    box-shadow: 0 20px 1px 20px $secondary;
}

// line
.status-box ul li.step-completed:before {
    background: $gray;
}

// dots
.status-box ul li.step-completed span:after {
    background: $gray;
}

// text
.status-box ul li.step-completed span {
    color: $gray;
}

// Current status
.status-box ul li.current span {

```

Pizza360

```
color: $primary;
}

.status-box ul li.current span:after {
  background: $primary;
}

.status-box ul li small {
  float: right;
  font-size: 14px;
}
```

main/resources/js/app.js

```
import axios from 'axios';
import Noty from 'noty'
import moment from 'moment'
import {initAdmin} from './admin'

let addToCart=document.querySelectorAll('.add-to-cart');
let cartCounter=document.querySelector('#cartCounter');

function updateCart(pizza){
  axios.post('/update-cart',pizza).then(res=>{
    console.log(res)
    cartCounter.innerText=res.data.totalQty
    new Noty({
      type:'success',
      timeout:1000,
      text: "Item added to the cart"
    }).show();
  })
}

addToCart.forEach((btn)=>{
  btn.addEventListener('click',(e)=>{
    let pizza=JSON.parse(btn.dataset.pizza);
    updateCart(pizza);
  })
})

//Remove alert message after x second
const alertMsg=document.querySelector('#success-alert')
if(alertMsg){
  setTimeout(()=>{
    alertMsg.remove()
  },2000)
}

initAdmin()
```

```

//change order status
let statuses = document.querySelectorAll('.status_line')
let hiddenInput = document.querySelector('#hiddenInput')
let order = hiddenInput ? hiddenInput.value : null
order = JSON.parse(order)
let time = document.createElement('small')

function updateStatus(order) {
  statuses.forEach((status) => {
    status.classList.remove('step-completed')
    status.classList.remove('current')
  })
  let stepCompleted = true;
  statuses.forEach((status) => {
    let dataProp = status.dataset.status
    if(stepCompleted) {
      status.classList.add('step-completed')
    }
    if(dataProp === order.status) {
      stepCompleted = false
      time.innerText = moment(order.updatedAt).format('hh:mm A')
      status.appendChild(time)
      if(status.nextElementSibling) {
        status.nextElementSibling.classList.add('current')
      }
    }
  })
}

updateStatus(order);

```

main/resources/js/admin.js

```

import axios from 'axios'
import moment from 'moment'
import Noty from 'noty'

export function initAdmin() {
  const orderTableBody = document.querySelector('#orderTableBody')
  let orders = []
  let markup

  axios.get('/admin/orders', {
    headers: {
      "X-Requested-With": "XMLHttpRequest"
    }
  }).then(res => {

```

```

orders = res.data
markup = generateMarkup(orders)
orderTableBody.innerHTML = markup
}).catch(err => {
  console.log(err)
})

function renderItem(items) {
  let parsedItems = Object.values(items)
  return parsedItems.map((menuItem) => {
    return `
      <p>${ menuItem.item.name } - ${ menuItem.qty } pcs </p>
    `
  }).join("")
}

function generateMarkup(orders) {
  return orders.map(order => {
    return `
      <tr>
        <td class="border px-4 py-2 text-green-900">
          <p>${ order._id }</p>
          <div>${ renderItem(order.items) }</div>
        </td>
        <td class="border px-4 py-2">${ order.customerId.name }</td>
        <td class="border px-4 py-2">${ order.address }</td>
        <td class="border px-4 py-2">
          <div class="inline-block relative w-64">
            <form action="/admin/order/status" method="POST">
              <input type="hidden" name="orderId" value="${ order._id }">
              <select name="status" onchange="this.form.submit()"
                class="block appearance-none w-full bg-white border border-gray-400 hover:border-
gray-500 px-4 py-2 pr-8 rounded shadow leading-tight focus:outline-none focus:shadow-outline">
                <option value="order_placed"
                  ${ order.status === 'order_placed' ? 'selected' : " }>
                  Placed</option>
                <option value="confirmed" ${ order.status === 'confirmed' ? 'selected' : " }>
                  Confirmed</option>
                <option value="prepared" ${ order.status === 'prepared' ? 'selected' : " }>
                  Prepared</option>
                <option value="delivered" ${ order.status === 'delivered' ? 'selected' : " }>
                  Delivered
                </option>
                <option value="completed" ${ order.status === 'completed' ? 'selected' : " }>
                  Completed
                </option>
              </select>
            </form>
          </div>
        </td>
      </tr>
    `
  })
}

```

```

    <div
      class="pointer-events-none absolute inset-y-0 right-0 flex items-center px-2 text-gray-
700">
      <svg class="fill-current h-4 w-4" xmlns="http://www.w3.org/2000/svg"
        viewBox="0 0 20 20">
        <path
          d="M9.293 12.951 7.071 15.657 8 14.141 10 10.828 5.757 6.586 4.343 8z"
        />
        </svg>
      </div>
    </div>
  </td>
  <td class="border px-4 py-2">
    ${ moment(order.createdAt).format('hh:mm A') }
  </td>
  <td class="border px-4 py-2">
    ${ order.paymentStatus ? 'paid' : 'Not paid' }
  </td>
</tr>
</table>
).join("")
}
}

```

main/routes/web.js

```

const homeController=require('../app/http/controllers/homeController')
const authController=require('../app/http/controllers/authController')
const cartController=require('../app/http/controllers/customers/cartController')
const orderController=require('../app/http/controllers/customers/orderController')
const AdminOrderController=require('../app/http/controllers/admin/orderController')
const StatusOrderController=require('../app/http/controllers/admin/statusController')

```

```
//middlewares
```

```

const guest=require('../app/http/middlewares/guest')
const auth=require('../app/http/middlewares/auth')
const admin=require('../app/http/middlewares/admin')

```

```
function initRoutes(app){
```

```

  app.get('/',homeController().index)
  app.get('/cart',cartController().index)
  app.get('/login',guest,authController().login)
  app.post('/login',authController().postLogin)
  app.get('/register',guest,authController().register)
  app.post('/logout',authController().logout)

```

```

app.post('/register',authController().postRegister)
app.post('/update-cart',cartController().update)
app.post('/orders',auth,orderController().store)
app.get('/customer/orders',auth,orderController().index)
app.get('/customer/orders/:id',auth,orderController().show)

app.get('/admin/orders',admin,AdminOrderController().index)
app.post('/admin/order/status',admin,StatusOrderController().update)

}

```

main/resources/views/layout.ejs

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?family=Oswald:wght@300&display=swap"
rel="stylesheet">
  <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
  <link rel="stylesheet" href="https://maxst.icons8.com/vue-static/landings/line-awesome/line-
awesome/1.3.0/css/line-awesome.min.css">
  <link rel="stylesheet" href="/css/app.css">
  <link rel="stylesheet" href="/scss/app.scss">
  <title>Document</title>
</head>
<body>
  <nav class="container mx-auto flex items-center justify-between py-4">
    <div>
      <a href="/"> </a>
    </div>
    <div>
      <ul class="flex items-center">
        <li class="ml-6"><a href="#">Menu</a></li>
        <li class="ml-6"><a href="#">Offers</a></li>
        <% if(user) { %>

        <li class="ml-6" >
          <form id="logout" action="/logout" method="POST">
            <a onclick="document.getElementById('logout').submit()" href="#">Logout</a></li>
          </form>
        <% }else{ %>
        <li class="ml-6" ><a href="/register">Register</a></li>

```

```

    <li class="ml-6"><a href="/login">Login</a></li>
    <% } %>
    <li class="ml-6"><a href="/cart" class="inline-block px-4 py-2 rounded-full flex items-
center">
        <span id="cartCounter" class="font-bold text-lg"><%=
session.cart?session.cart.totalQty:"%></span>
        </a></li>
    </ul>
</div>
</nav>
<%- body %>
<script src="/js/app.js"></script>
</body>
</html>

```

main/resources/views/layout.ejs

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Oswald:wght@300&display=swap"
rel="stylesheet">
    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://maxst.icons8.com/vue-static/landings/line-awesome/line-
awesome/1.3.0/css/line-awesome.min.css">
    <link rel="stylesheet" href="/css/app.css">
    <link rel="stylesheet" href="/scss/app.scss">
    <title>Document</title>
</head>
<body>
    <nav class="container mx-auto flex items-center justify-between py-4">
        <div>
            <a href="/"> </a>
        </div>
        <div>
            <ul class="flex items-center">
                <li class="ml-6"><a href="#">Menu</a></li>
                <li class="ml-6"><a href="#">Offers</a></li>
                <% if(user) { %>

```

```

<li class="ml-6" >
  <form id="logout" action="/logout" method="POST">
    <a onclick="document.getElementById('logout').submit()" href="#">Logout</a></li>
  </form>
  <% }else{ %>
  <li class="ml-6" ><a href="/register">Register</a></li>
  <li class="ml-6"><a href="/login">Login</a></li>
  <% } %>
  <li class="ml-6"><a href="/cart" class="inline-block px-4 py-2 rounded-full flex items-
center">
    <span id="cartCounter" class="font-bold text-lg"><%=
session.cart?session.cart.totalQty:"%></span>
    </a></li>
  </ul>
</div>
</nav>
<%- body %>
<script src="/js/app.js"></script>
</body>
</html>

```

resources/views/customers/singleOrder.ejs

```

<section class="status">
  <div class="container mx-auto">
    <div class="status-box w-full lg:w-2/3 mx-auto">
      <div class="flex items-center justify-between mb-12">
        <h1 class="text-xl font-bold">Track delivery status</h1>
        <h6 class="bg-white py-1 rounded-full px-4 text-green-600 text-xs"><%=
order._id %></h6>
        <input id="hiddenInput" type="hidden" value="<%= JSON.stringify(order) %>">
      </div>
      <ul>
        <li class="status_line text-sm md:text-xl pb-16" data-status="order_placed"><span>Order
Placed</span>
        </li>
        <li class="status_line text-sm md:text-xl pb-16" data-status="confirmed"><span>Order
confirmation</span>
        </li>
        <li class="status_line text-sm md:text-xl pb-16" data-
status="prepared"><span>Preparation</span></li>
        <li class="status_line text-sm md:text-xl pb-16" data-status="delivered"><span>Out for delivery
</span>
        </li>
        <li class="status_line text-sm md:text-xl" data-

```



```

status="completed"><span>Complete</span></li>
</ul>
</div>
</div>
</section>

```

main/resources/views/customers/orders.ejs

```

<section class="orders light-section">
  <div class="container mx-auto pt-12">
    <h1 class="font-bold text-lg mb-4">All Orders</h1>
    <% if(messages.success) { %>
      <div id="success-alert" class="flex items-center bg-green-500 text-white text-sm font-bold px-4 py-3"
role="alert">
        <svg class="fill-current w-4 h-4 mr-2" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20
20">
          <path
            d="M12.432 0c1.34 0 2.01.912 2.01 1.957 0 1.305-1.164 2.512-2.679 2.512-1.269 0-2.009-
.75-1.974-1.99C9.789 1.436 10.67 0 12.432 0zM8.309 20c-1.058 0-1.833-.652-1.093-3.52411.214-
5.092c.211-.814.246-1.141 0-1.141-.317 0-1.689.562-2.502 1.117l-.528-.88c2.572-2.186 5.531-3.467
6.801-3.467 1.057 0 1.233 1.273.705 3.231l-1.391 5.352c-.246.945-.141 1.271.106 1.271.317 0 1.357-.392
2.379-1.2071.6.814C12.098 19.02 9.365 20 8.309 20z" />
          </svg>
          <p><%= messages.success %></p>
        </div>
      <% } %>
      <table class="w-full table-auto bg-white">
        <thead>
          <tr>
            <th class="px-4 py-4 text-left">Orders</th>
            <th class="px-4 py-4 text-left">Phone</th>
            <th class="px-4 py-4 text-left">Address</th>
            <th class="px-4 py-4 text-left">Time</th>
          </tr>
        </thead>
        <tbody>
          <% if(orders.length) { %>
            <% orders.forEach((order)=>{ %>
              <tr>
                <td class="border px-4 py-2">
                  <a class="link" href="/customer/orders/<%=order._id %>"><%= order._id %></a>
                </td>
                <td class="border px-4 py-2">
                  <%= order.phone %>
                </td>

```

```

        <td class="border px-4 py-2">
            <%= order.address %>
        </td>
        <td class="border px-4 py-2">
            <%= moment(order.createdAt).format('hh:mm:A') %>
        </td>
    </tr>
    <% }) %>
    <% } else { %>
        <tr>
            <td class="p-4"><span>No orders found</span></td>
        </tr>
    <% } %>
</tbody>

</table>
</div>
</section>

```

Conclusion

We have made a online pizza ordering website, named PIZZ360 which is in partial fulfilment state we have done extensive hard work in the front end section and also created a login signup page whose information is stored in a database on the server, we have use various web development technologies till now ,like HTML5 ,CSS3, Bootstrap , JavaScript , Mongo DB, NodeJS and various tools like Visual studio code , Google chrome, this is all possible because of the guidance of our mentors , further we will add backend to make it fully functional .

As someone with no prior experience in web development, we believe our time spent in training and discovering new languages was well worth it and contributed to finding an acceptable solution to an important aspect of web design and development.

Future Scope

In the coming future we aim to make our website more applicable and well appearing ,we will add payment mode to this website so that customers can order their pizza in just few clicks and every information will be saved at the backend.

References

- ★ <https://www.w3schools.com/>
- ★ <https://stackoverflow.com/>
- ★ <https://github.com/KunalSingh777/Pizz360>
- ★ www.bootstrap.com
- ★ www.youtube.com
- ★ <https://github.com/>