TRAINING REPORT

WEB DEVELOPMENT

Submitted in partial fulfillment of the

Requirements for the award of

Degree of Bachelor of Technology in Computer Science & Engineering

Submitted By-

Name: Divyaansh

Enrollment Number: 03514802718

Submitted To-

Department of Computer Science & Engineering

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY ROHINI, DELHI

CERTIFICATE

PEPCODING EDUCATION (OPC) PRIVATE LTD.

3rd Floor, 15, Vaishali, Pitampura, New Delhi-110034

Website: www.pepcoding.com Phone: +911 4019 4461

DATE: 8th July 2020

TO WHOM IT MAY CONCERN

This is to certify that Divyaansh, B.Tech student at Maharaja Agrasen Institute of Technology, Rohini has successfully completed his 10 week training in our 'Web Development' course from 21st March 2020 to 30th May 2020. We found him sincere in his work and well-coordinated with his colleagues.

We wish him the best of luck for his future endeavors.

Sumeet

Sumeet Malik

Director

DECLARATION

I hereby declare that the Training Report entitled "Web Development" is an authentic record of my work as requirements of 10 weeks Training during the period from March 21, 2020, to May 30, 2020, for the award of the degree of B.Tech (Computer Science & Engineering), GGSIPU, under the guidance of Mr. Jasbir Singh.

Name: Divyaansh

Enrollment No.: 03514802718

Date: _____

Certified that the above statement made by the student is correct to the best of our knowledge and belief.

Signatures

Examined by:

1. **2.**

(Guide/Trainer) **(Faculty Coordinator)**

Head of Department

(Signature and Seal)

ACKNOWLEDGEMENT

The successful completion of this training report would not have been possible without the support and assistance of many individuals and organizations. I feel immensely blessed to have gotten this during my training program. I would like to take this opportunity to offer my earnest admiration to each one of them.

First and foremost, I am highly indebted to Mr. Sumeet Malik (Founder of Pepcoding*)* who took confidence in me and provided me with the training at Pepcoding.com. I had a wonderful and unforgettable experience being part of such a lovely and lively team. I am also grateful to Mr. Jasbir Singh (Mentor of Web development at Pepcoding), who taught me for 10 weeks and helped me in my project. Without his constant guidance and suggestions, this report would have been nowhere near completion. My gratitude for his trust and generosity goes beyond words.

Name: Divyaansh

Enrollment No.: 03514802718

ABOUT THE COMPANY

Pepcoding Education Private Ltd. is a private self - funding teaching institute. It is a start-up by Mr. Sumeet Malik that was established on December 6, 2017. With a team of excellent members each with their weight working towards a common goal – to impact the way computer science is learned and taught. It trains multiple students from multiple backgrounds in the field of Data Structures and Algorithms in Java, C++, and Web Development.

Table of Contents

List of Tables

List of Figures

CHAPTER 1: Introduction

1.1. About Summer Training

The summer training at PepCoding Education was for Web Development.

The course content is:-

1.1.1. Basic JavaScript

- Basic syntax and Memory map
- Synchronous and asynchronous coding
- Callback functions and callback hell
- Closure
- Https requests (get and post)
- Creating server and streaming files
- CLI interface (using readline module)
- Web sockets
- Promises and Async await

1.1.2. Frontend

- Introduction to HTML5 and CSS3
- Introduction to Pug
- HTML elements and CSS selector
- Responsive Design(media tag)
- Version Control through Git
- Deployment on Git
- Bootstrap
- Dom manipulation using canvas

1.1.3. Backend

- Introduction to npm packages, package.json
- Server creation through Express.js
- MVC architecture
- Axios introduction and usage
- Mongo database introduction
- Integration of MongoDB with Nodejs

1.1.4. React.js

- Introduction to react.js.
- Class based components and function based components
- States and Props
- Introduction to hooks (useState())
- Lifecycle methods (componentDidMount() , etc)
- Css module
- Form validation
- Sending https requests
- Adding Routes

1.2. Training Objectives

- Understand the principles of creating an effective web page, including an in-depth consideration of information architecture.
- Develop skills in analyzing the usability of a website.
- Learn techniques of responsive web design, including media queries.

- Develop basic programming skills using JavaScript and jQuery.
- Learn HTML and CSS.
- Understand how searching on the web works.

CHAPTER 2: Tools and Technologies Used

2.1. Technology Used

2.1.1. HTML

- HTML stands for Hypertext Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

2.1.2. CSS

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on the screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

2.1.3. Node.js

- Node.js is an open-source server environment
- Node.js is free

- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

2.1.4. Express.js

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of the Express framework −

- Allows users to set up middleware to respond to HTTP Requests.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

2.1.5. MongoDB

MongoDB is an open-source document database and a leading NoSQL database. MongoDB is written in C++. This tutorial will give you a great understanding of MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

2.1.6. Pug.js

Pug.js is an HTML templating engine, which means you can write a much simpler Pug code, which the Pug compiler will compile into HTML code, that browser can understand.
Pug has powerful features like conditions, loops, includes using which we can render HTML code based on user input or reference data. Pug also supports JavaScript natively, hence using JavaScript expressions, we can format HTML code.

2.2. Tools Used

2.2.1. Google Chrome (Web Browser)

A Web Browser is a software program that allows a user to locate, access, and display web pages. It is based on the Chrome V8 engine with the support of tools like console, testing tools, etc.

Web browsers are used primarily for displaying and accessing websites on the internet, as well as other content created using languages such as Hypertext Markup Language (HTML) and Extensible Markup Language (XML).

2.2.2. Visual Code Studio (IDE)

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs.

2.2.3. Postman (to test API)

Postman is a popular API client that makes it easy for developers to create, share, test, and document APIs. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses.

2.2.4. MongoDB Atlas (To manage database easily)

MongoDB is an open-source document database and a leading NoSQL database. MongoDB is written in C++.

MongoDB Atlas is the cloud version of MongoDB.

It is used to store all the information about our plans and users.

2.2.5. GitHub

GitHub is a Git repository hosting service, but it adds many of its features. While Git is a command-line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as wikis and basic task management tools for every project.

CHAPTER 3: Technical Content

3.1. Overview of the Project

The “PeppersDev Food Delivery” project is based on the Node.js environment which harnesses specific NPM packages and modules. Every package installation is done on a macOS based system with NPM as the package manager in a secure system environment. This makes the project distribution friendly and easy management.
The following is the overall contemplating schema upon which my project stands upon:

I. Frameworks

- **Node.js -** The active runtime environment based on ECMA6 Javascript scripting language and Chrome V8 Javascript Engine.

- **Express.js -** Minimalist Web framework fabricated to be used with Node.Js Applications. It can handle multiple different Http requests at a specific URL. Express is an open-source and flexible web app framework designed to make developing websites, web apps, & API's much easier.

- **Pug.js -** Pug.js is an HTML templating engine, which means you can write a much simpler Pug code, which the Pug compiler will compile into HTML code, that the browser can understand. Pug has powerful features like conditions, loops, includes, mixins using which we can render HTML code based on user input or reference data. Pug also supports JavaScript natively, hence using JavaScript expressions, we can format HTML code.

- **MongoDB -** MongoDB is a document database with the scalability and flexibility option. It is a NoSQL based database system that works upon Binary JSON (BSON).

II. Application Program Interface(s) (APIs)

- **Stripe Payment Gateway API -** Stripe is a development-oriented secure payment and transaction solution that is popular for online businesses and scalable startups.

III. NPM Modules

- **Nodemailer:** for sending mail to the user whenever he/she sign up and when he/she forgets the password.

- **Bcrypt:** for password encryption in the project using simple predefined algorithms.

- **Crypto:** Generation of random tokens for authentication during Password resetting.

- **Mongoose:** Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks.

IV. Middlewares

- **Helmet:** secures the Express apps by setting various HTTP headers.

- **Express-rate-limiter:** limit repeated requests to public APIs and/or endpoints such as password reset.

- **Express-mongo-sanitize:** sanitization done to malicious users could send an object containing a $ operator, or including a, which could change the context of database operation.

- **Cookie Parser:** parses cookies attached to the client request object.

V. Testing Tools

- **Postman:** Postman is a REST client used for performing backend API testing. In this we test all kinds of requests i.e. get request, post request, patch request, delete request. In case of a get request, we just need to pass the URL of the API and then it will fetch all the information which is sent by that API from the backend. In the case of the post request, we need to pass the URL of the API and with that, we also have to pass the information, which we want to write in the database in the form of JSON.

3.2. Purpose of the Project

The project aims to simulate a consumer-based food delivery application to learn and use all the required technical packages and software programs. The project is a food delivery app with a user management database and content designing. The app has the following purposes:

- Create a database consisting of multiple users.
- Order meals and track the order and wishlist activity against every user present in the database.
- Pay for the meal you ordered with the help of an integrated payment system made using Stripe API
- Authentication and authorization of the users
- Integrating some security features into the application using the basic javascript packages and modules
- Manage JSON data of every user

3.3. Project Architecture

My project follows the traditional **Model-View-Controller (MVC)** design pattern.

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller. Each of these components is built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects.

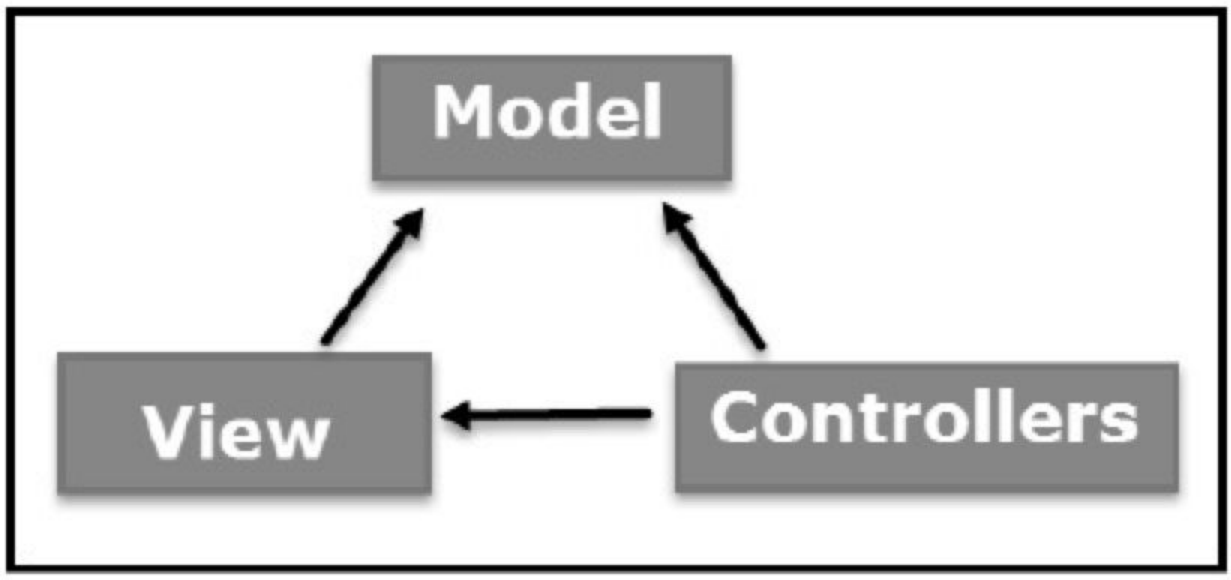

Fig. 3.1: MVC model 1

Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it, and update its data back to the database or use it to render data.

View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component, and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

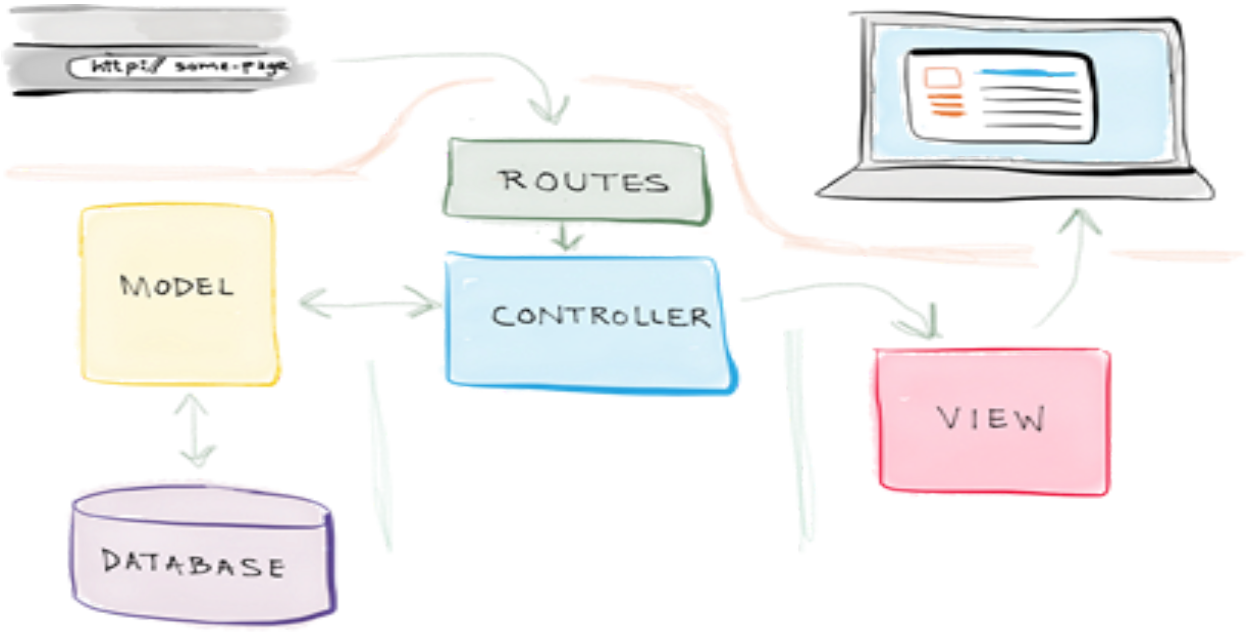

Fig. 3.2: MVC model 2

A request is made — say, when a user enters a URL associated with your application. A **route** associated with that URL maps the URL to a controller action. That **controller** action leverages the necessary **model**(s) to retrieve information from the database and then passes that data off to a view. And that **view** renders the final page.

3.4. Use-Case of the project

This is a Food Ordering website in which a user can order meals and pay for it online. In this website, there are two roles i.e. admin, the user. This project utilizes an asynchronous threading model that can handle multiple users simultaneously at a particular time which makes the project scalable in the future.

3.4.1. User

A user is an entity that consumes a product or employs non-admin service.

The user first has to sign up/login on the website to buy meals. If the user visited the website for the first time then the user should signup on the website but if the user already registered then the user should log in to the website to buy the meals.

The user can wishlist the meals if the user wants to buy it later. Otherwise, if the user wants to buy the meal straightaway, then the user will be transferred to a new window to place the order by paying the money through debit/credit card. After buying the meal, that meal will be added to his previous order page and the user can see what meals he ordered previously.

If by any chance, the user forgets the password then the user can click on forget password button on his login window and then the user has to type his registered email id and then a mail is sent to that email id with a given token, to reset the password, the user should type that token and the new password. Once the token is verified, the new password will be set.

If the user wants to change his/her personal information, then the user must be logged in to the website. After the user is logged in, the user can change his/her information. If the user wants to change his password while he/she is logged in, then the user can simply click to the change password button and provide his/her old password and then can type his new password.

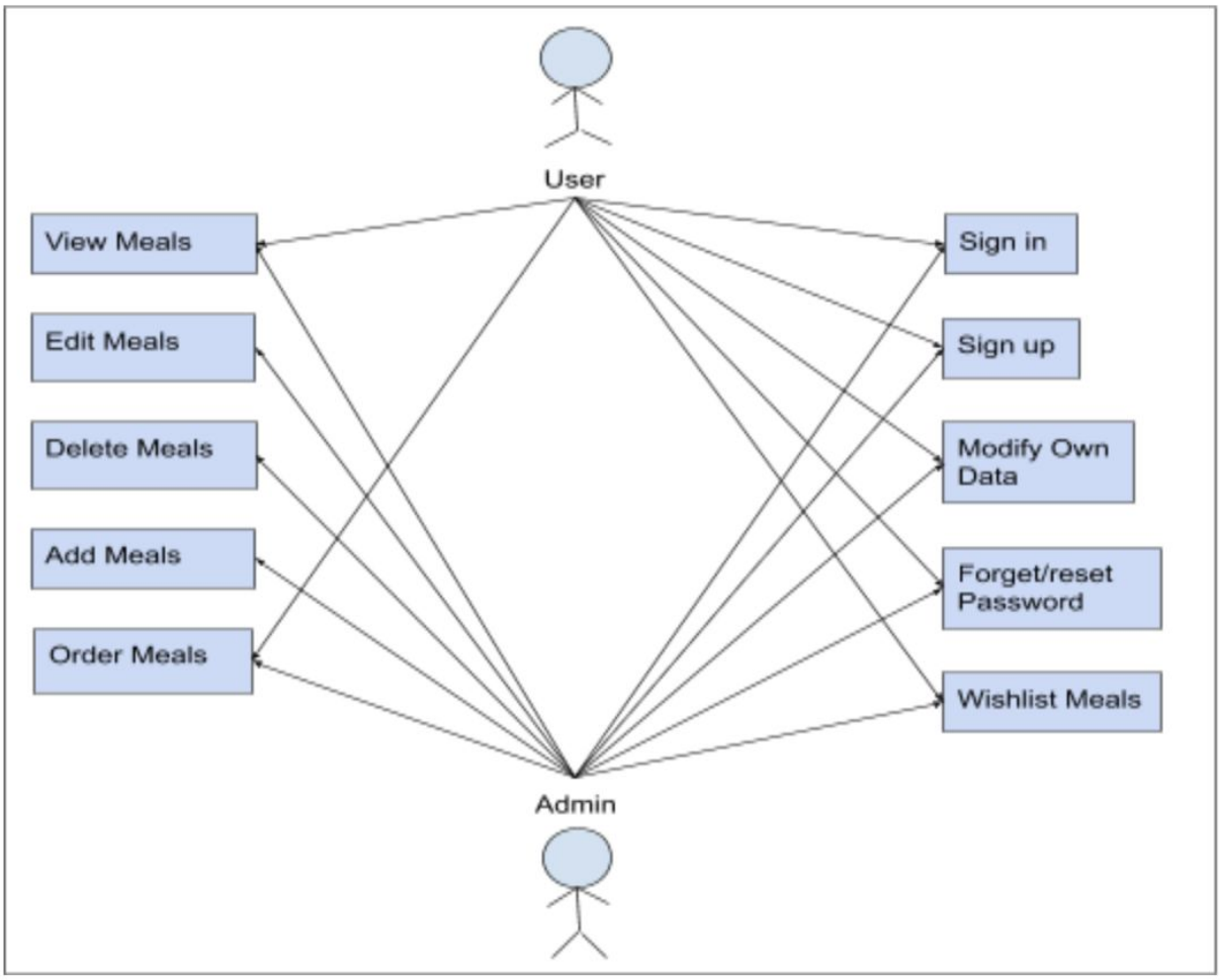

Fig. 3.3: Use Case Diagram

3.4.2. Admin

An admin is a superuser that has modification access to the project.

The admin can do everything the user can. But additionally, the admin has controls over the meals. The admin can add, edit, delete the meals. To edit the meals, the admin can change the price, change the content of the meals.

3.5. Implementation of the Project

The whole project can be explained with 3 main stages that define the implementation of the project.

Stage 1: Idea evaluation, data collection
Stage 2: Creating UI for the project(Frontend development)
Stage 3: Creating and Testing APIs for the project(Backend development)
Stage 4: Integrating Frontend and Backend of the project

3.5.1. Idea evaluation and data collection

The project was initialized with the proposal of the project. The training institute helped in getting the mainframe of the project. With the avid situations of COVID19, food delivery situations suffered a lot at the time. It was the simulation of real-time food delivery services like Swiggy, etc. But the project is not a replica of those, rather works on a different design of the application.

Rough drafting of the project was done to calculate the requirements of the project and software packages on paper. With that in mind, appropriate data were collected from the internet, querying for images, names, etc.

3.5.2. Development: APIs and Web App

The development of the project started in the 2nd week of the training period.
This required a knowledge of server creation, infrastructure management, and setting up the project. Following are the steps used in the whole development process:

1. Setting up the Server: Created a server using express. Deployed a local server to test and modify settings in a real-time environment.

2. Wireframing of the project: With the help of the calculations done for the requirements of data and source collection, objects were framed and wireframed accordingly.
   Initially, 3 pages were composed: Main landing page, sign in page and order checkout Page. They have connected accordingly with the appropriate order. More pages were created as per the development of the whole project.

3. Designing UI: A minimalistic approach was used to plan the design of the interface that is easy on the eyeballs. Metro colors, the arrangement of placeholders and input forms, subtle placement of pictures and icons, and other things were done to make this project look simpler yet easy to understand and use.

4. The shift was done from the frontend towards the backend. The backend was much simpler to understand as the frontend was created first. API was planned and programmed for the project. The backend was created with the help of HTTP modules, express, and mongoose. Things are discussed in further points.

5. Created functions for plan and user which can add, delete, get, modify the plan's, and user's object.

6. Adding routes to the functions of plan and user so that the functions will be ready to test. Routing is essentially done to optimize page navigation and asynchronous user management.

7. Testing of the APIs was done with the help of Postman. The application was connected to postman via the desktop interface and call requests were made via it to test the response time and JSON delivery. The API performance was recorded in the postman and a report was generated. The report was used to analyze the performance of the application.

8. Authentication and Authorisation: The application was configured with features like signup, login, protect route(to check whether the user is logged in or not), authorize(to check whether the user is admin or not), forget and reset the password. This was done with the help of NPM modules and packages. Everything was connected with the backend.

- The *forget password* feature was integrated into the sign-in page itself. It asks the email ID of the registered user and sends a token of authentication to the same. Upon entering the token into the next form, the token is validated and matched with the user's permanent saved token. If it is valid, the user is redirected to a reset password page where he/she can set a new password for authentication used for the subsequent sessions of the application.

- The *protect route* function checks whether the user is logged in or not because some features can be accessed if the user is logged in, without that he/she cant access those features. To do so, I take the help of an npm package i.e. jsonwebtoken. The jwt token is saved in the form of a cookie in the user's browser when the user signup or login. And when the user logout from his/her account, the jwt cookie expires exactly at that time so that the protect route function can check if the jwt token is present which means the user is logged in and if it is not, it means that the user is not logged in.

9. Final Testing: After connecting all the authentication backend, the whole application was tested again with the help of Postman.
    - Postman has inbuilt functions to run scripts in different environments. The project is tested to run in different versions of browsers and cross-platform applications.
    - It also has an easy storage solution that is used to store the temporary JSON of the form to test via the API.

10. After the completion of the Backend of the project, the whole frontend which was created previously in HTML then converted into templates using Pug.js which is a templating engine and the reason for this conversion is because in Pug file we can use javascript easily without any hassle as compared to HTML. This helped the seamless integration of the backend with the frontend.

11. After the completion of all the templates using Pug.js, a file was created, named frontend.js which I put in the public folder. In this file, I write codes to redefine the events like click event, the on Change event. I fetch the details entered by the user in the form after which that button was clicked, and make that information in form of JSON and send it to the backend while we make an Axios request to that API.

12. The project was completed and thoroughly revised again for any unintentional bugs or errors.

Chapter 4: Snapshots

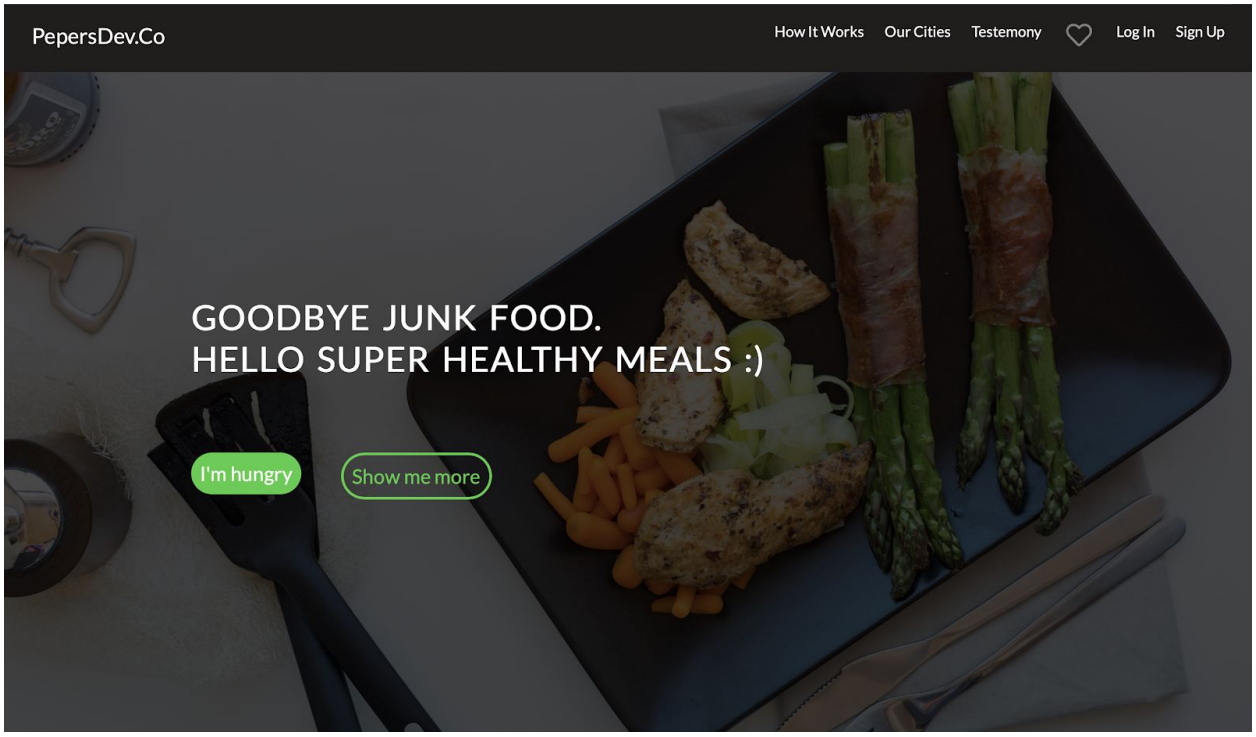

Fig. 4.1: Front Page

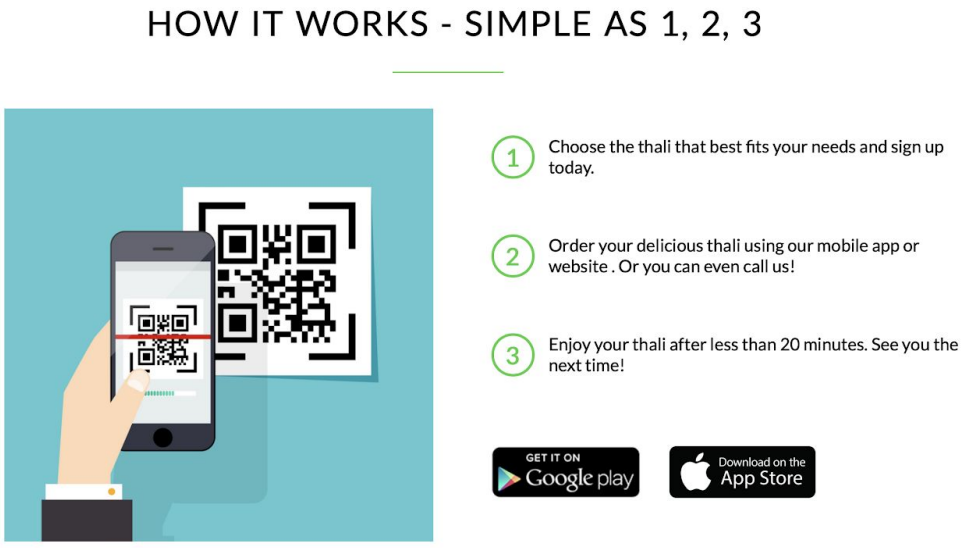

Fig. 4.2: How it works Page

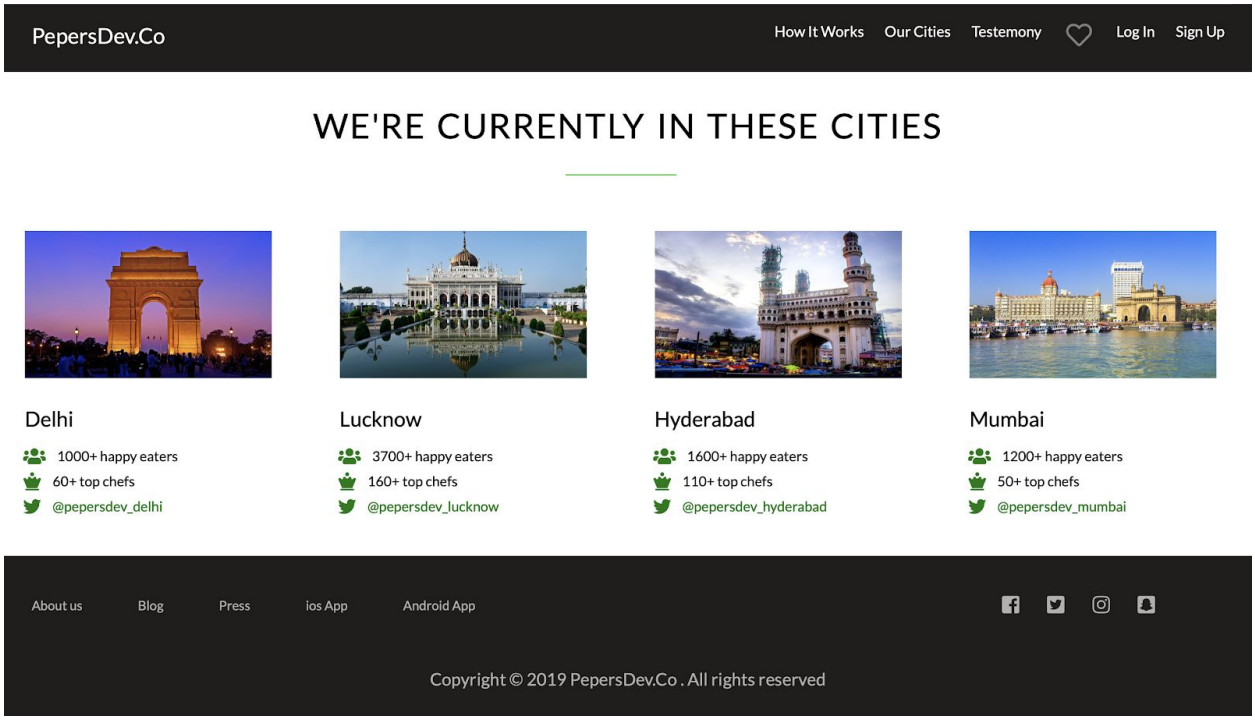

Fig. 4.3: Cities Page

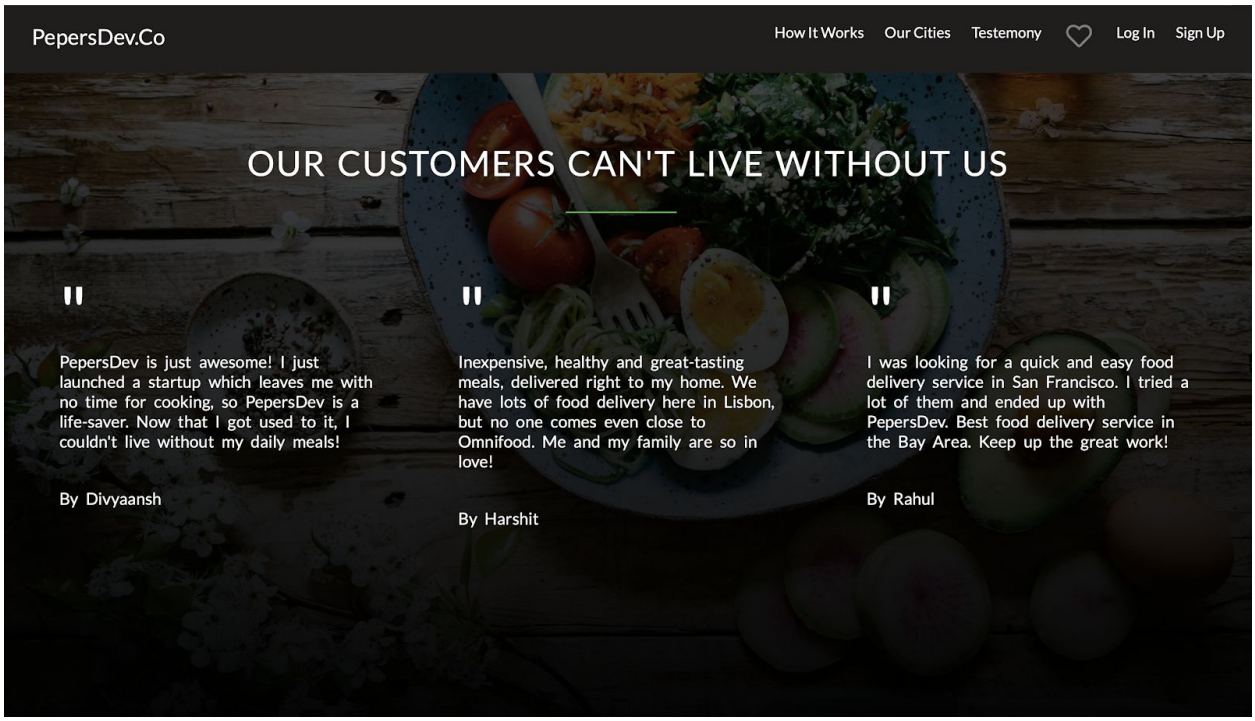

Fig. 4.4: Review Page

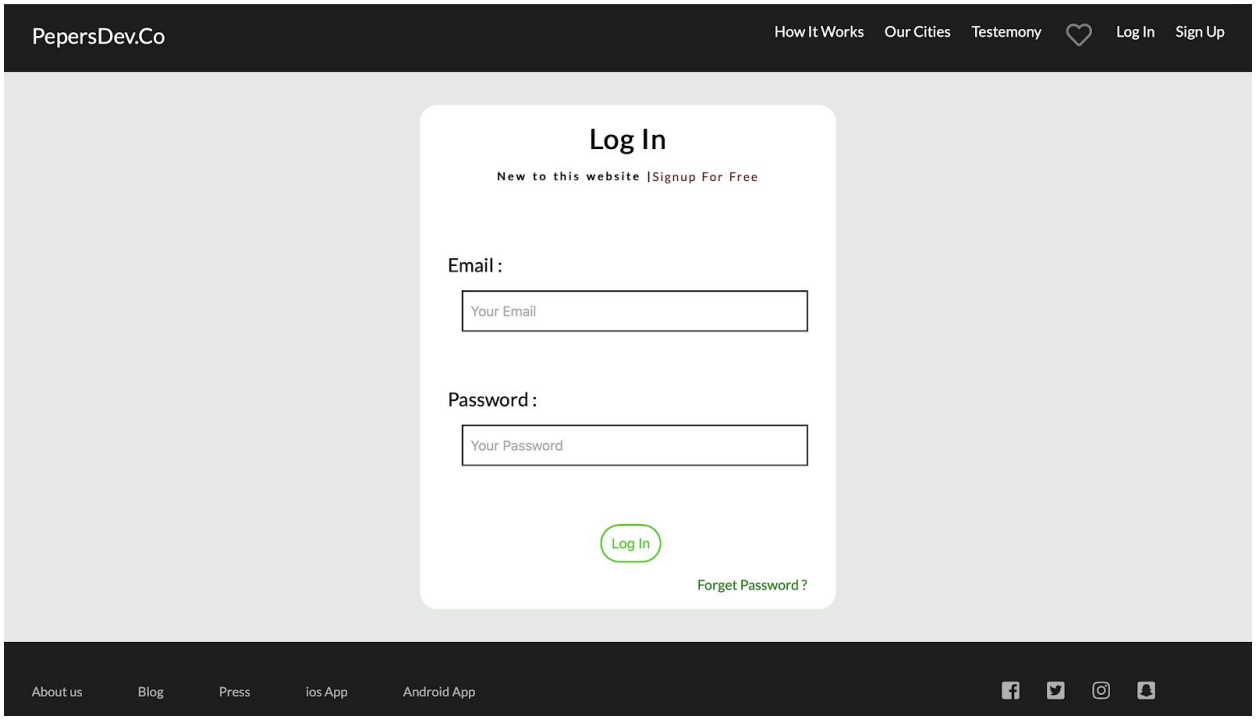

Fig. 4.5: Login Page

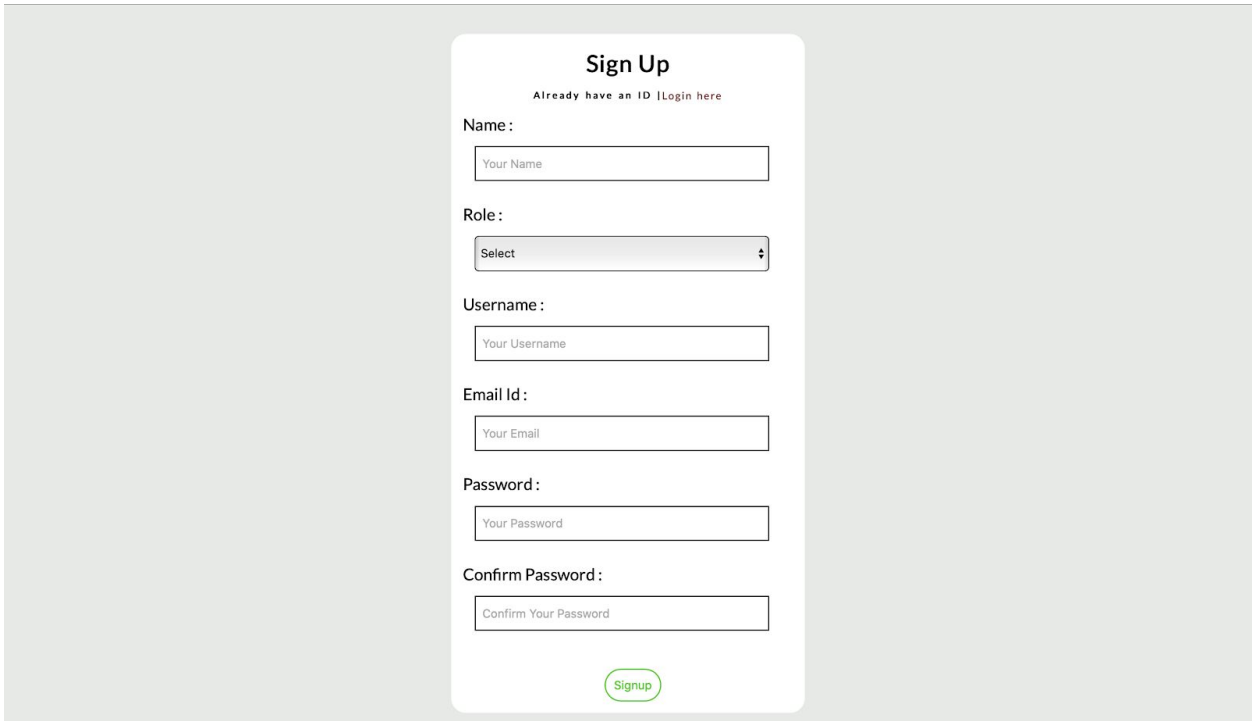

Fig. 4.6: Signup Page

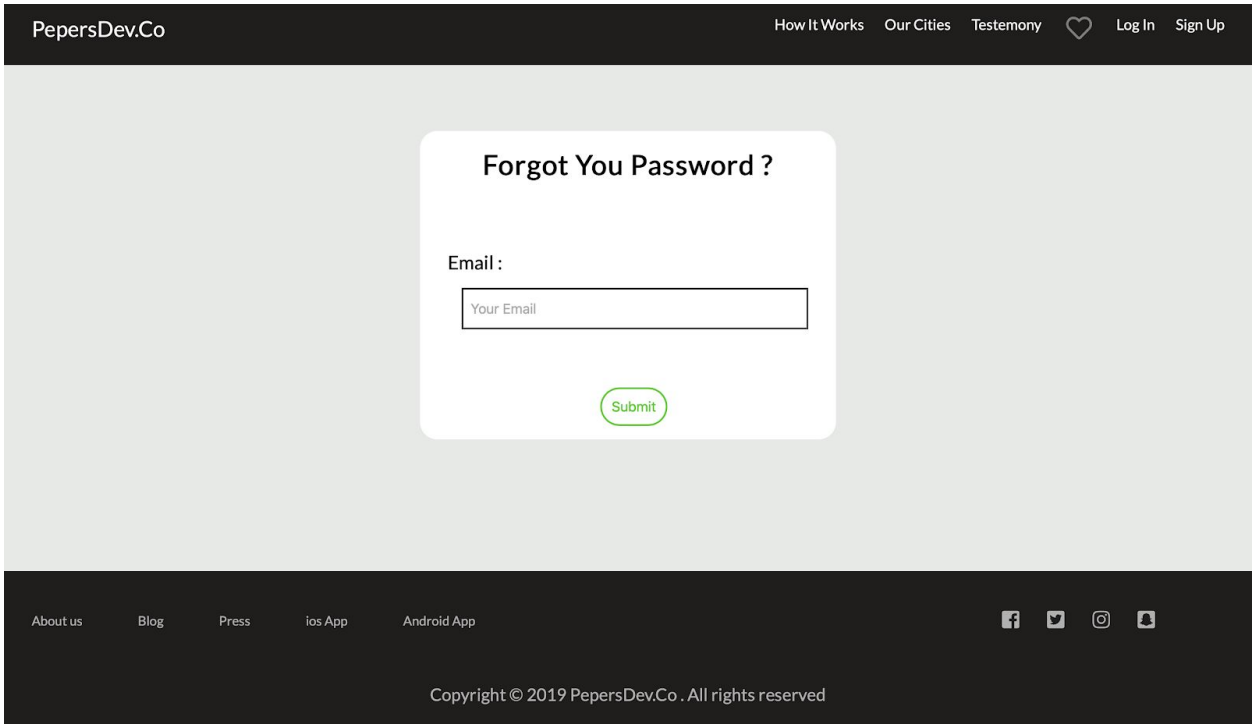

Fig. 4.7: Forget Password Page

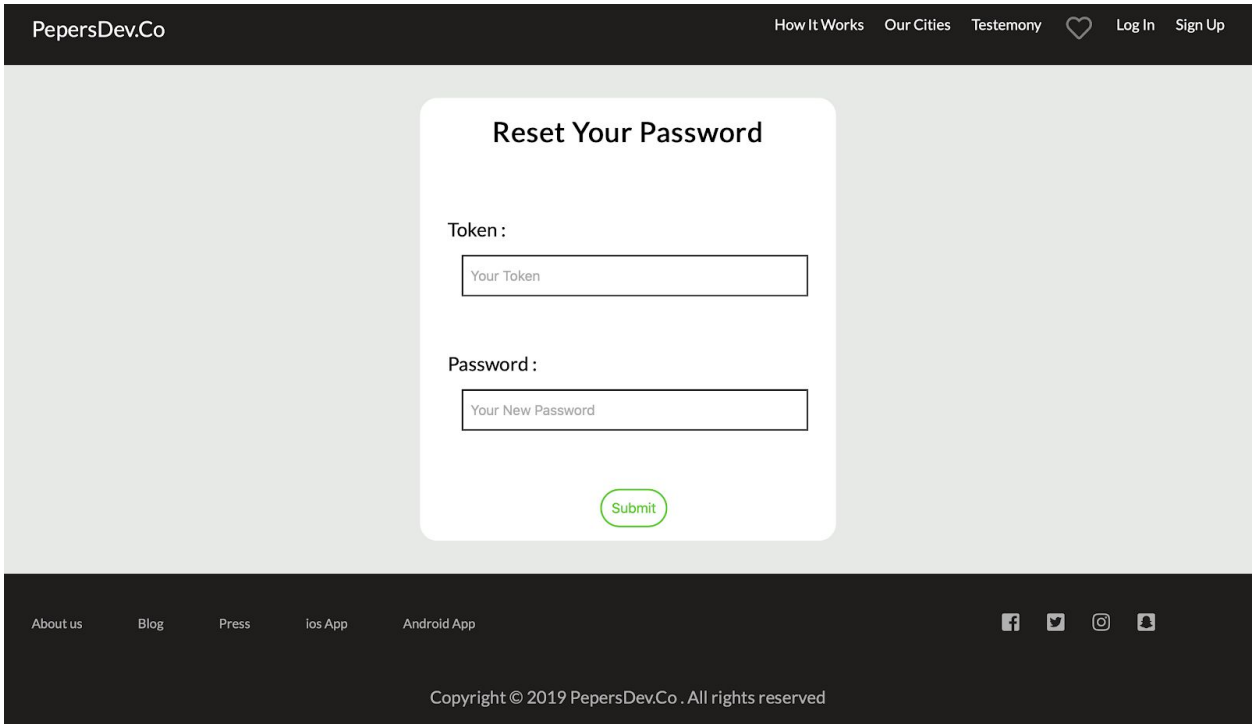

Fig. 4.8: Reset Password Page

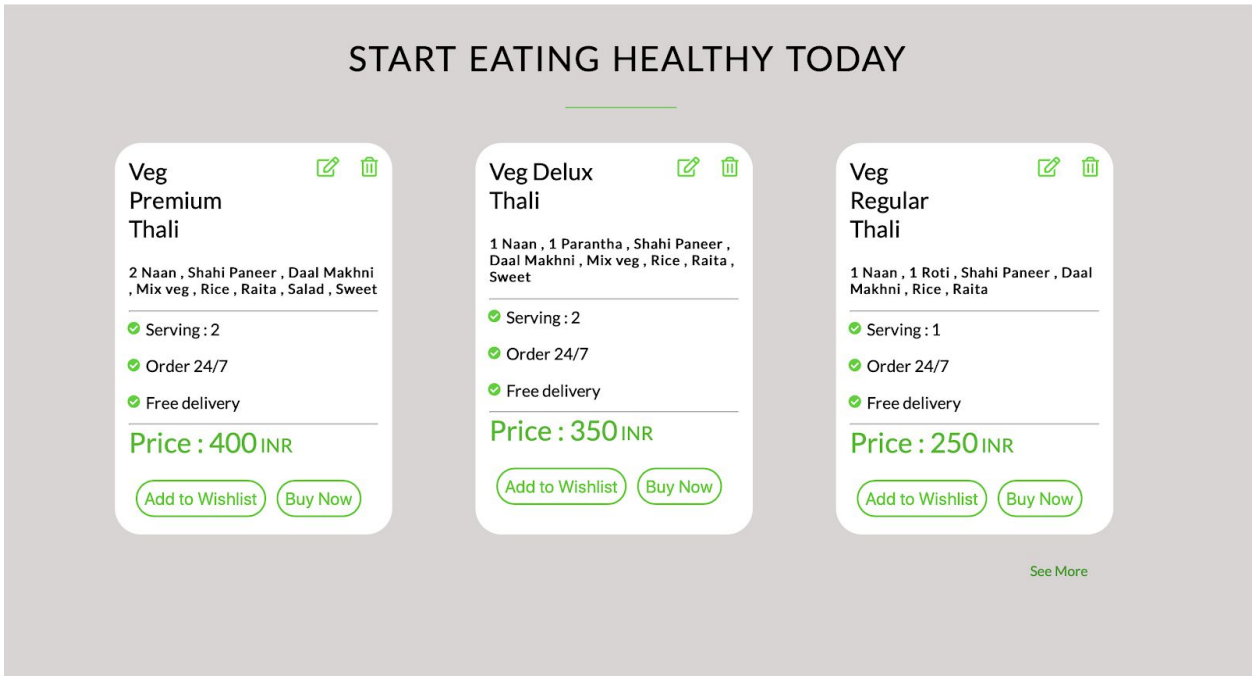

Fig. 4.9: Top 3 Plans Page

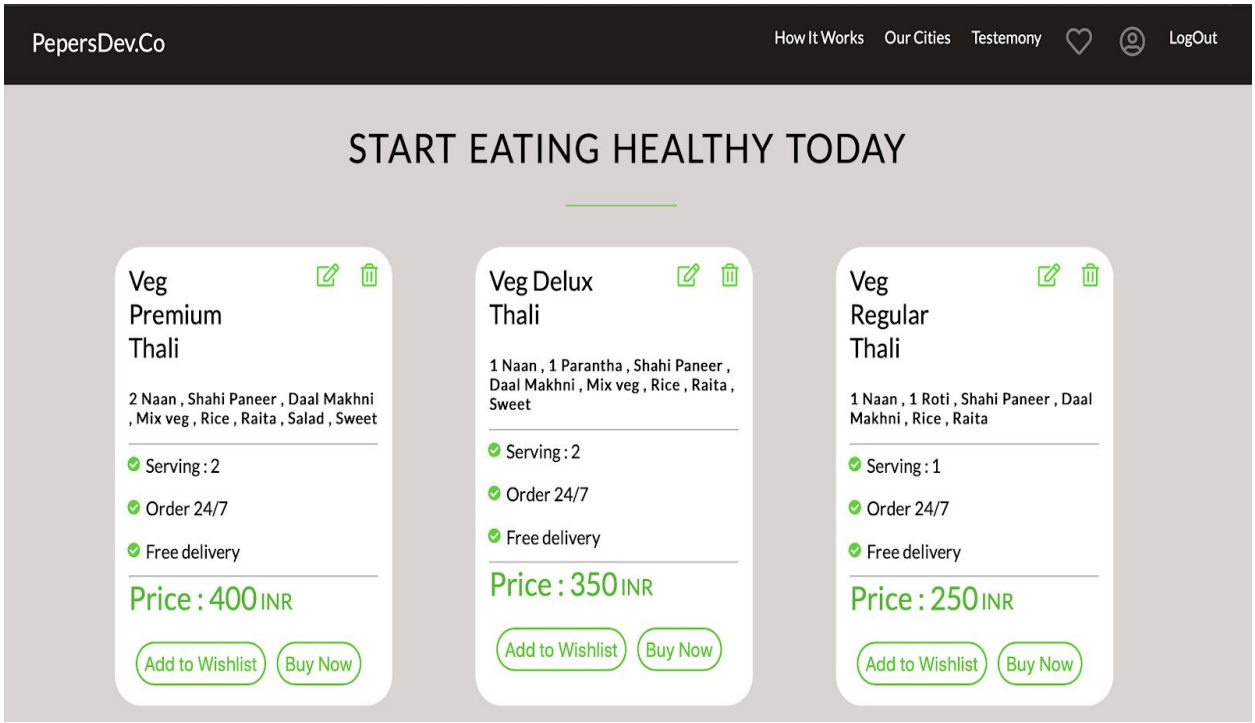

Fig. 4.10.1: Plans Page 1

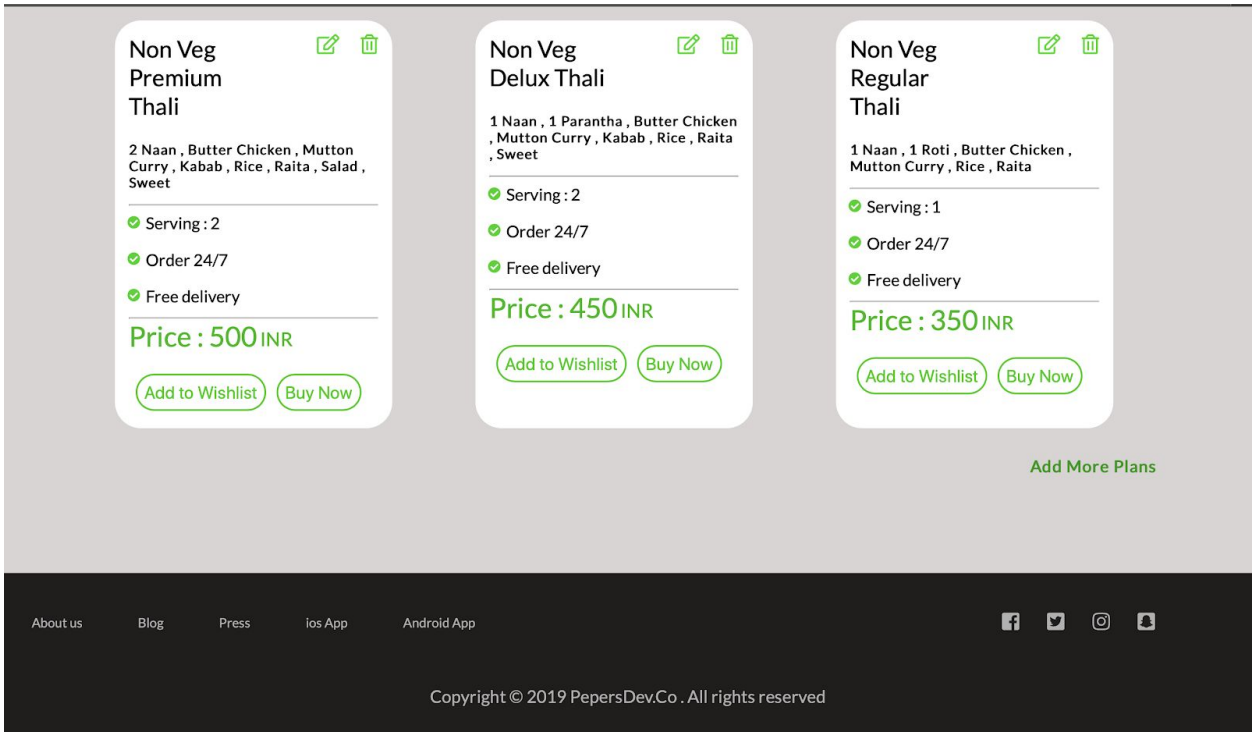

Fig. 4.10.2: Plans Page 2

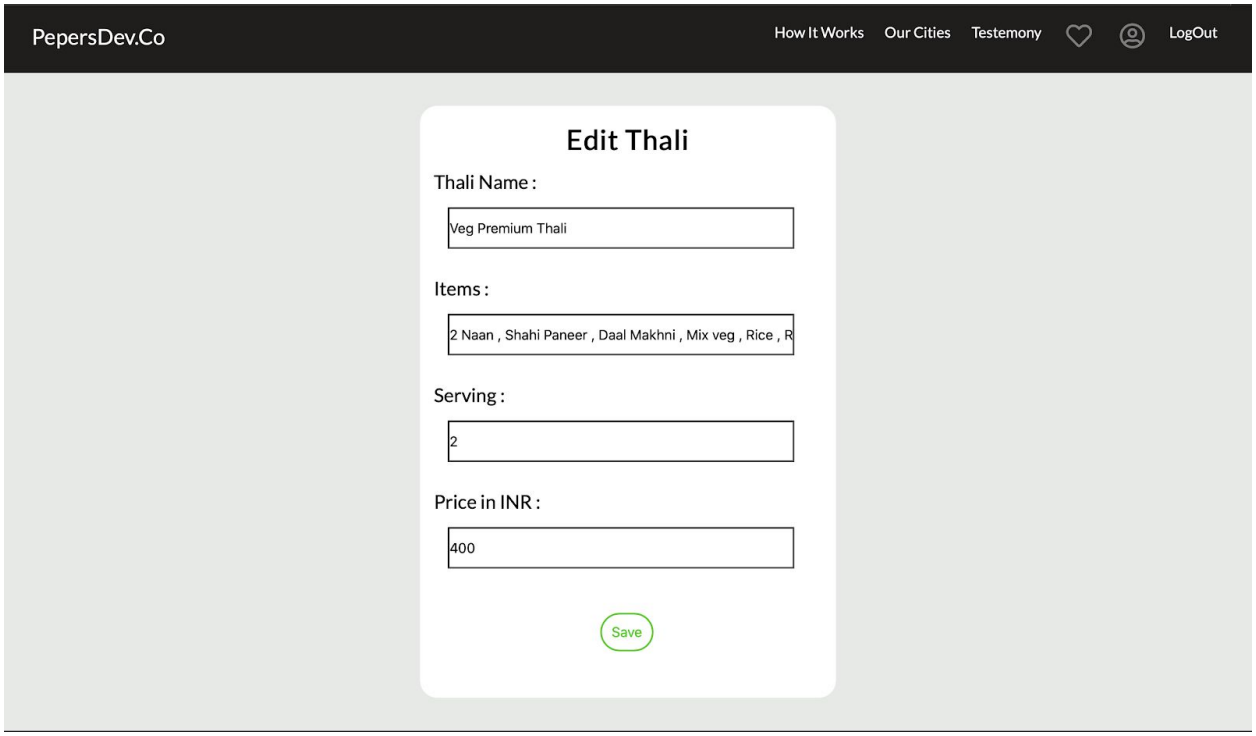

Fig. 4.11: Edit Thali/Plans Page

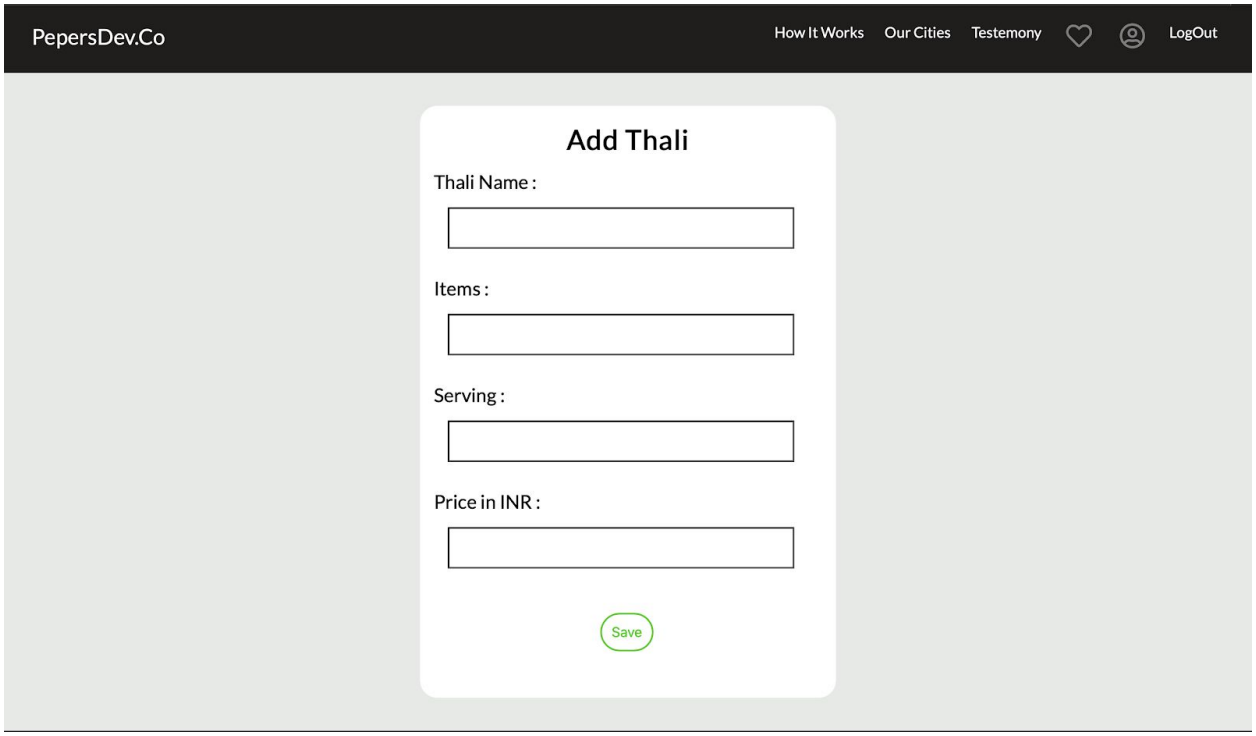

Fig. 4.12: Delete Thali/Plan Page

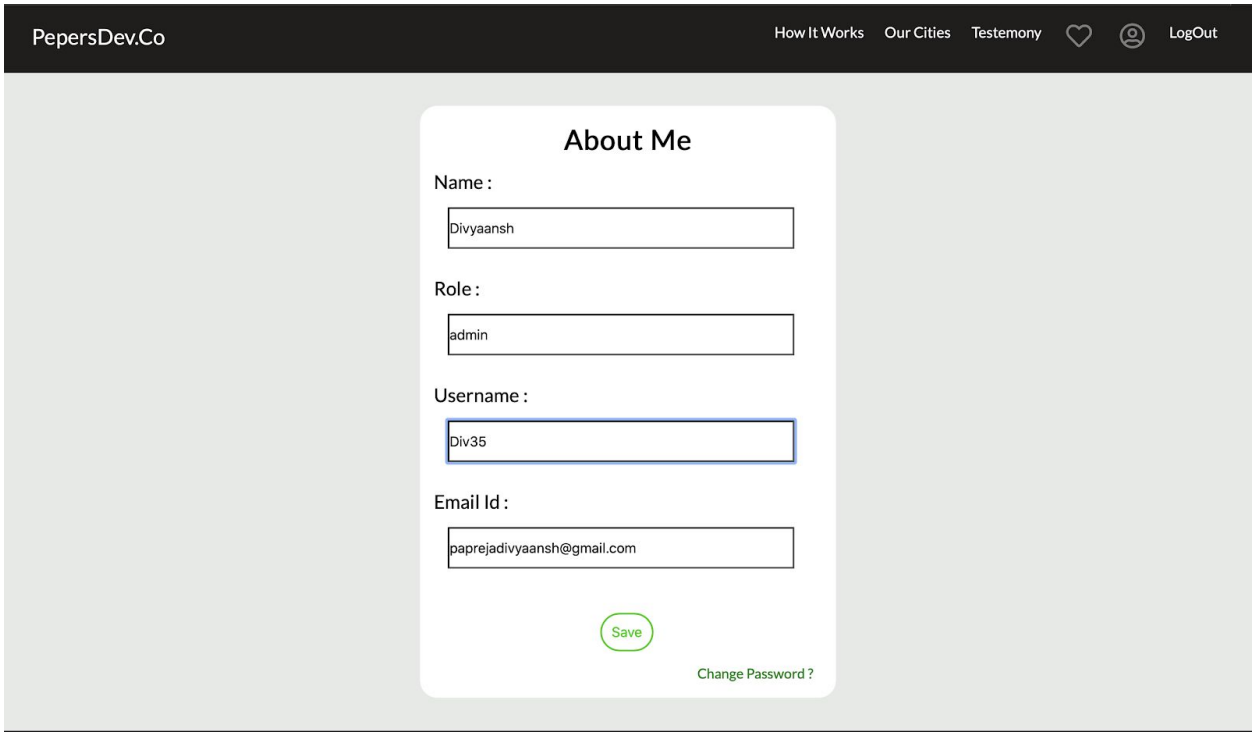

Fig. 4.13: About Page

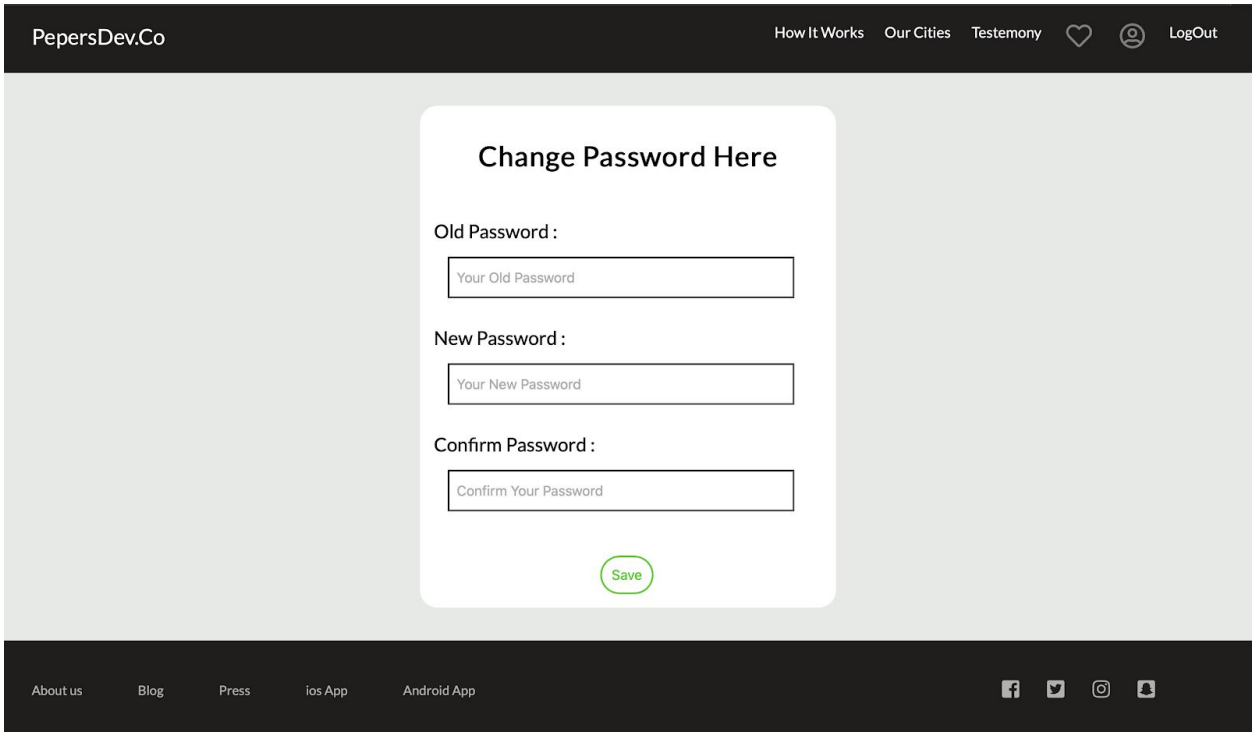

Fig. 4.14: Change Password Page

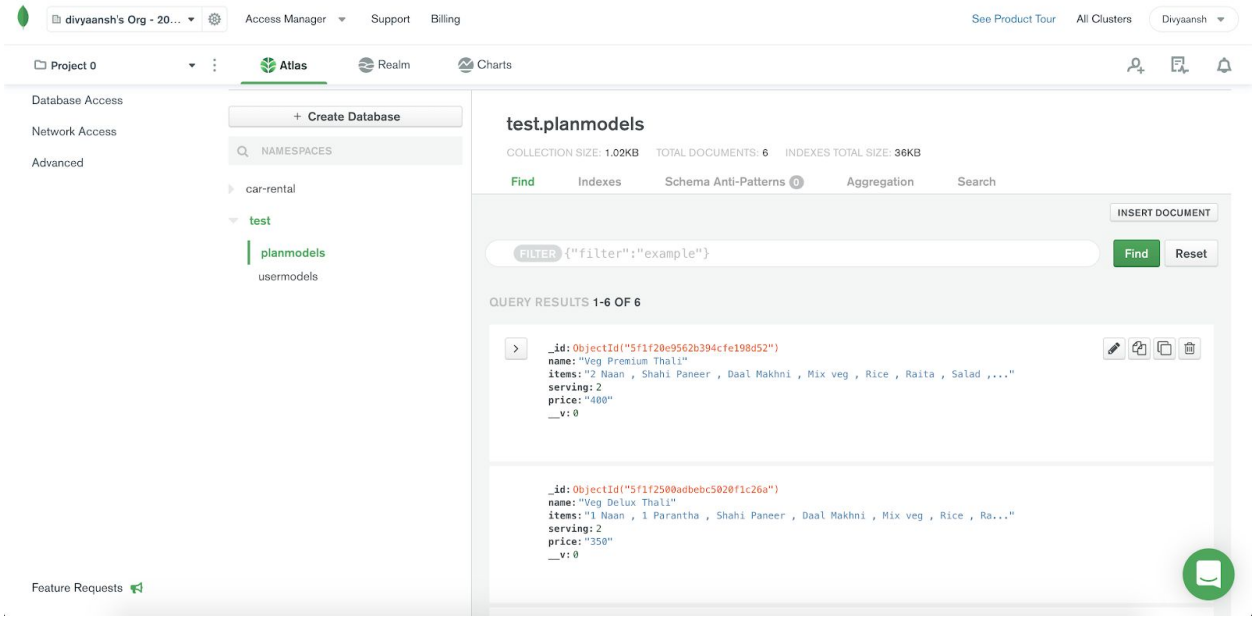

Fig. 4.15: Plan's Database Page in MongoDB

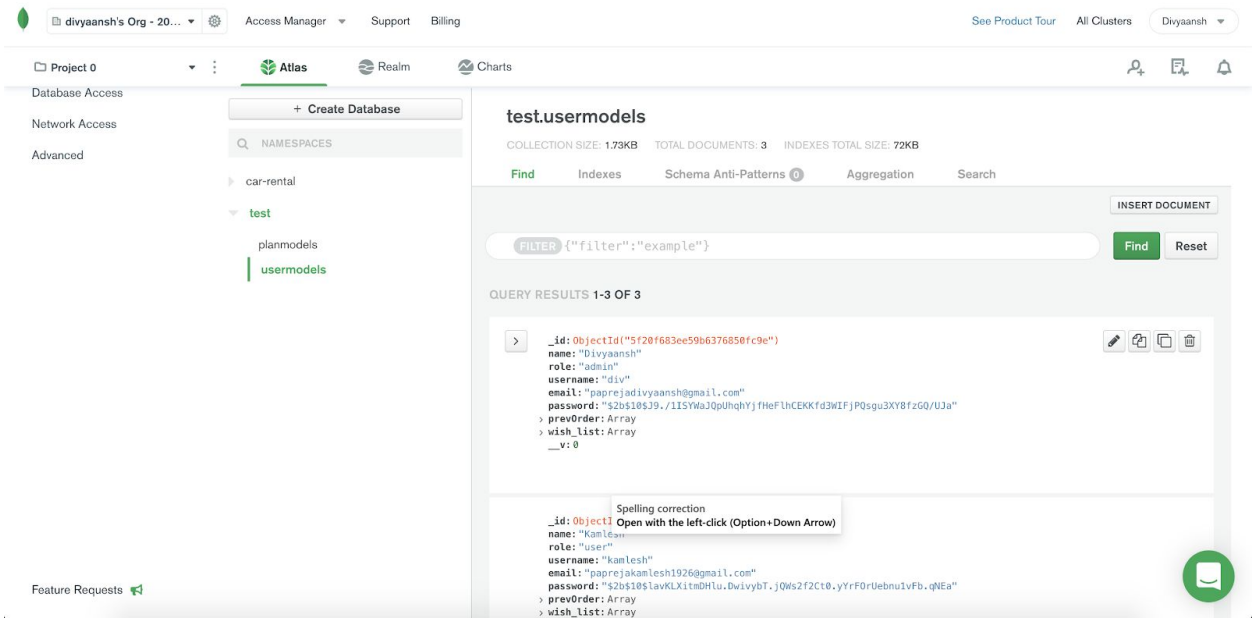

Fig. 4.16: User's Database Page in MongoDB

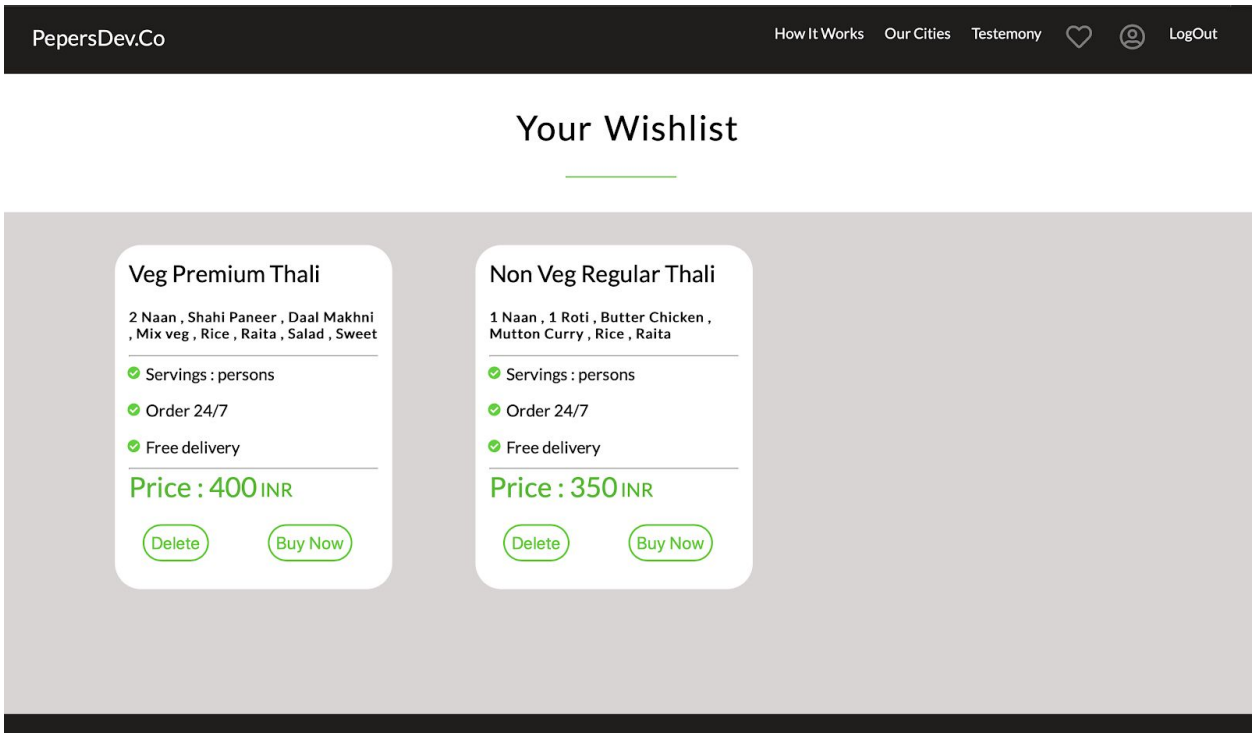

Fig. 4.17: Wishlist Page

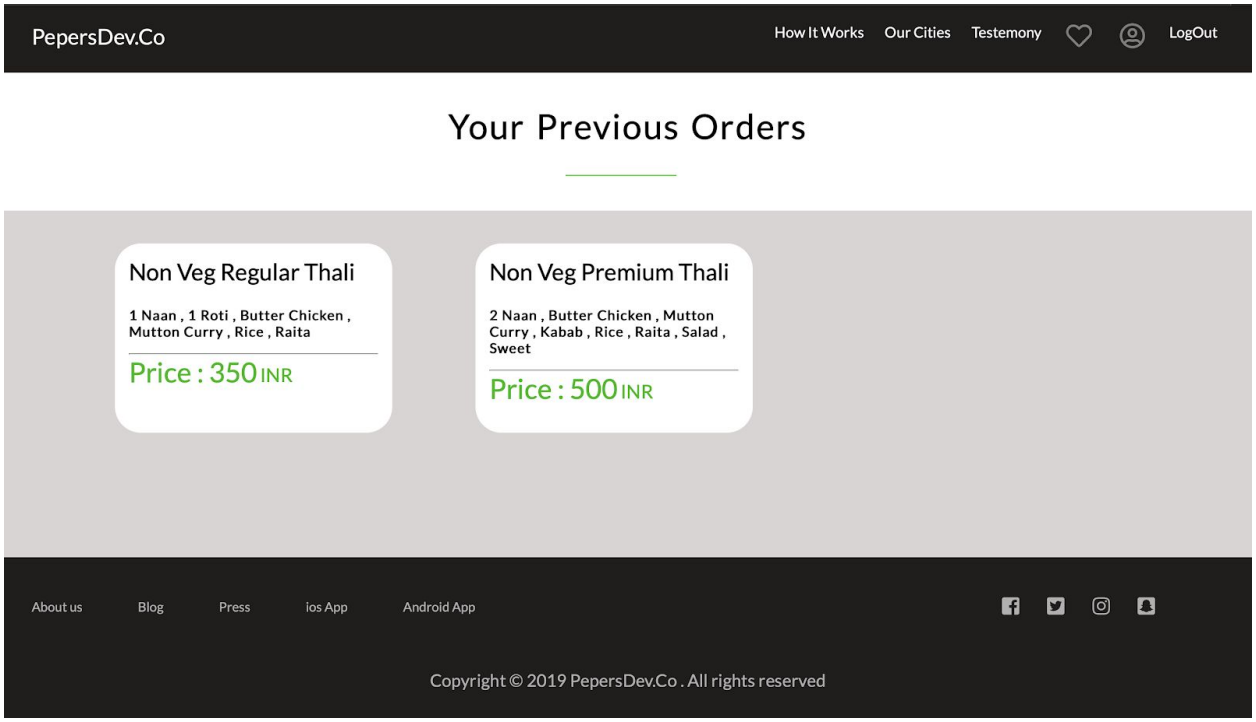

Fig. 4.18: My Orders Page

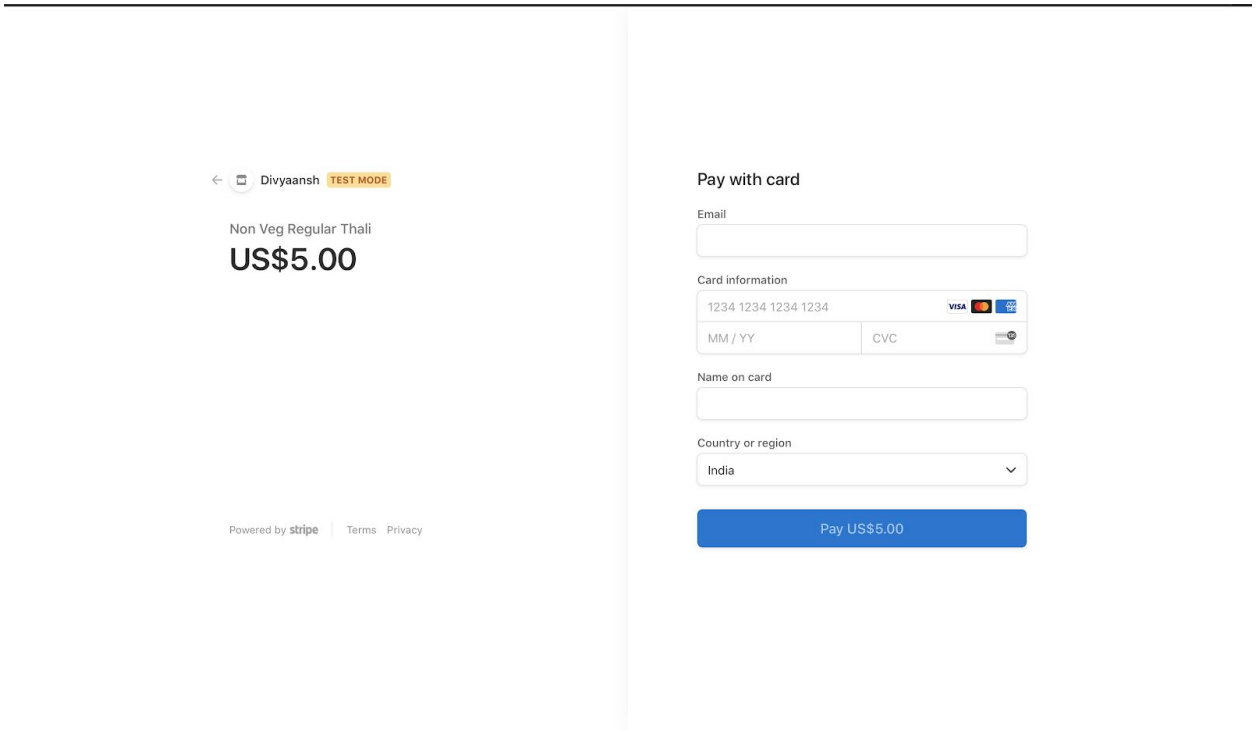

Fig. 4.19: Payment Page

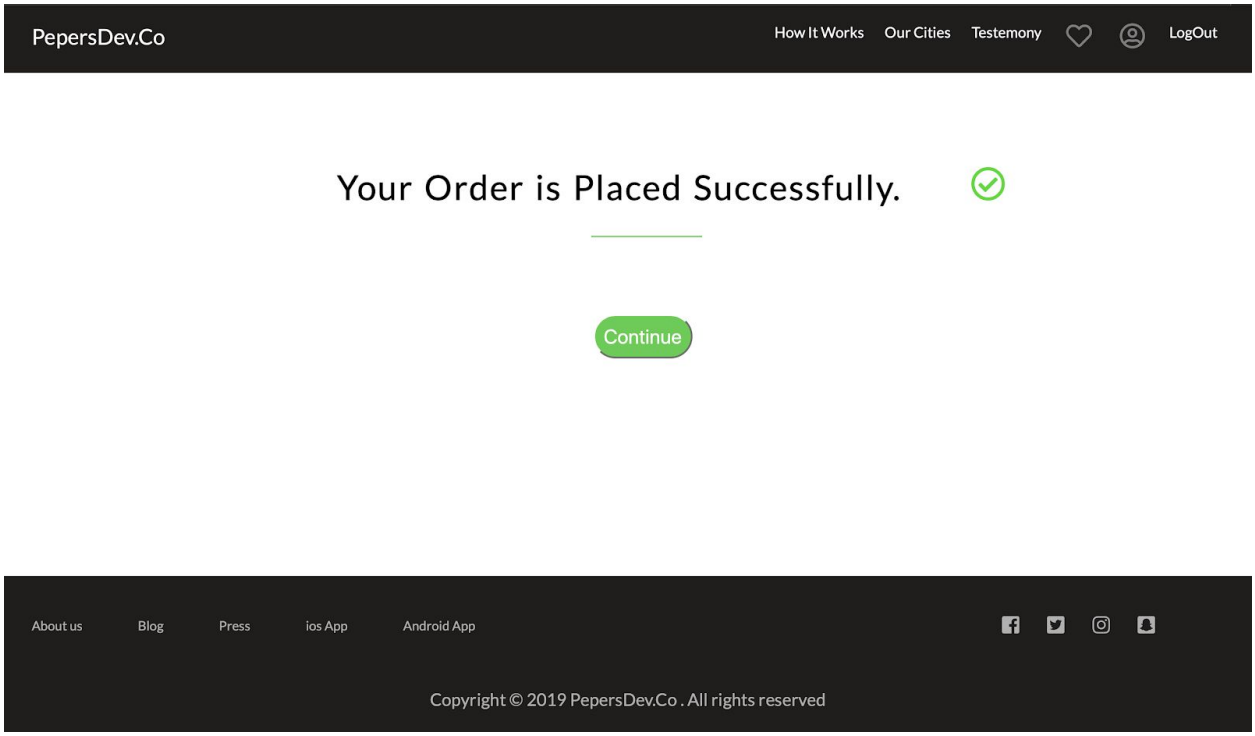

Fig. 4.20: Order Placed Successfully Page

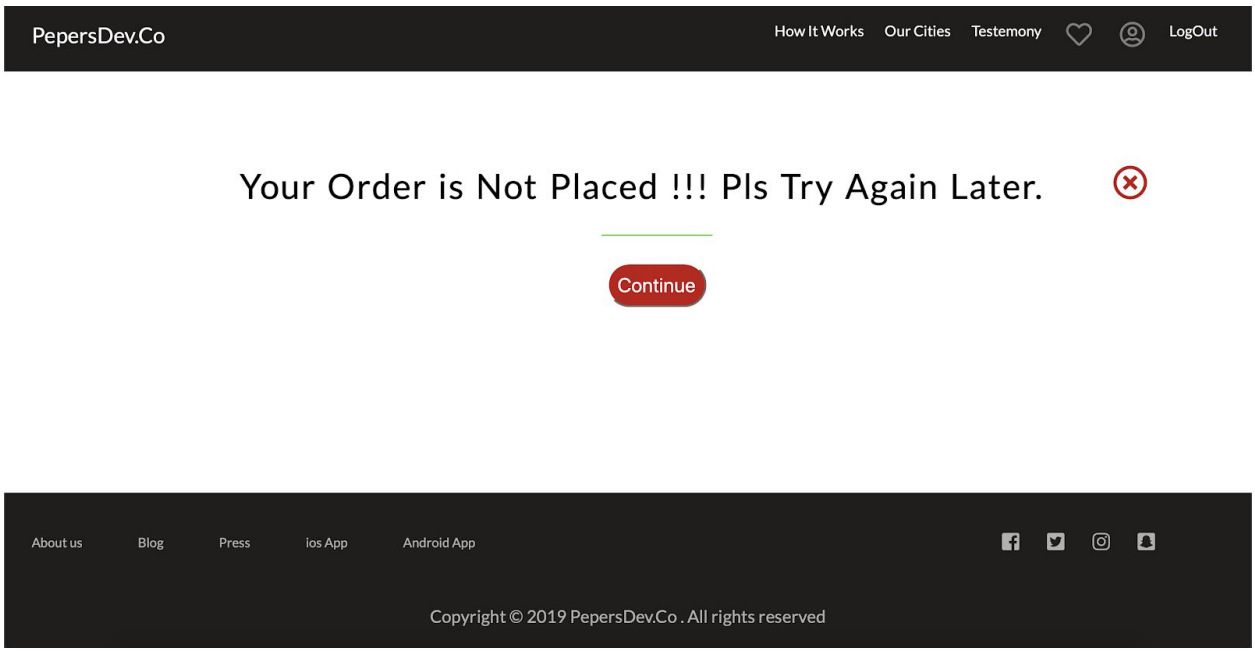

Fig. 4.21: Order Not Placed Page

CHAPTER 5: Results and Discussion

The whole coursework is done in an appropriate timeframe of the training period and with respected deadlines for the project implementation. The objective of the training was duly fulfilled to learn about web technology and gain hands-on experience in web development. All the relevant concepts of web technology were studied in particular and applied via the practical problem given by the training institution. The confidence is gained in the following technological skill:

1. Database Management using MongoDB
2. Node.JS Server + Express.JS
3. Templating Engine
4. NPM Packages
5. Security in applications
6. Frontend UX/UI Designing
7. Working with an API
8. Payment Gateway

CHAPTER 6: Conclusion and Future Scope

In the 10 weeks of summer training at Pepcoding Education Limited, we were taught about the complete fundamentals of full-stack web development. This included both frontend basics such as HTML, CSS, Pug.js, React.js basics, etc, and also backend fundamentals which included but weren't limited to MongoDB, Node.js, Express.js, etc.

"PepersDev.co - A Food Delivery System" is a food delivery website that allows users to register themselves on the system, choose food delivery plans that they might need in their day-to-day, and get them very conveniently. It has an intuitive interface that allows users to securely access and modifies their data. It will immensely help restaurants to optimize and control their restaurants. It will also increasingly help maintain the restaurant's functions effectively and accurately and also reduce the use of manual entries. This software helps food orders to be maintained as records in the system. It maintains prepper records in the database system.

CHAPTER 7: Weekly Jobs Summary

Week: 1

Description of Activity	Time Spent
Basic Javascript syntax eg. console.log(), basics of the array. File system function like fs.mkdirSync(), etc	6 hrs
Different Data types, Object and Arrays functions, how to declare functions, and call them.	4 hrs
Introduction to Npm packages	2 hrs

Table 7.1

List one thing that went particularly well this week

File system functions and objects

List one thing that was the most challenging this week

Introduction to Npm packages

Self Evaluation: A+

Week: 2

Description of Activity	Time Spent
Synchronous codes like fs.readfileSync() and use them.	2 hrs
Callback functions and Callback hell, Asynchronous codes	8 hrs
Closures	2 hrs

Table 7.2

List one thing that went particularly well this week

Synchronous codes

List one thing that was the most challenging this week

Closures

Self Evaluation: A

Week: 3

Description of Activity	Time Spent
How requests work, Https requests, How to send get requests, and post requests.	4 hrs
Creating Servers, Introduction to Express.js, Streaming images and videos on servers	4 hrs
Readline Modules and Introduction to the Event emitter	4 hrs

Table 7.3

List one thing that went particularly well this week

Creating servers

List one thing that was the most challenging this week

Event Emitter

Self Evaluation: A

Week: 4

Description of Activity	Time Spent
Web sockets and a little project on it.	4 hrs
Promises and Async await	6 hrs
A little CLI based project for Weather Api using all javascript knowledge till now.	4 hrs

Table 7.4

List one thing that went particularly well this week

Async Await

List one thing that was the most challenging this week

Web socket project and Promises

Self Evaluation: A+

Week: 5

Description of Activity	Time Spent
Introduction to HTML and there different tags like div, span, h1, h2, etc.	3 hrs
Introduction to CSS and Bootstrap, A small static web page.	2 hrs
A Short project of Whiteboard using HTML Canvas and CSS	7 hrs

Table 7.5

List one thing that went particularly well this week

Introduction to HTML

List one thing that was the most challenging this week

HTML Canvas

Self Evaluation: A+

Week: 6

Description of Activity	Time Spent
Introduction to Jquery	3 hrs
Creating Basic Frontend UI for the Project	5 hrs
Introduction to Node.js, Express.js, Introduction, and installation of MongoDB in the system, Practice commands of MongoDB on terminal	4 hrs

Table 7.6

List one thing that went particularly well this week

Introduction to MongoDB

List one thing that was the most challenging this week

Nothing

Self Evaluation: A+

Week: 7

Description of Activity	Time Spent
Integration of MongoDB with Express.js and writing sample code for testing its proper working.	3 hrs
Creating Basic API's for the project and testing them through Postman	9 hrs

Table 7.7

List one thing that went particularly well this week

Everything

List one thing that was the most challenging this week

Nothing

Self Evaluation: A+

Week: 8

Description of Activity	Time Spent
Adding Basic Authentication to the Project	5 hrs
Introduction to Pug.js, Syntax of Pug.js, Conversion of HTML and Pug.js	4 hrs

Table 7.8

List one thing that went particularly well this week

Authentication

List one thing that was the most challenging this week

Why we use Pug.js

Self Evaluation: A

Week: 9

Description of Activity	Time Spent
Converting the UI made in HTML previously to Pug.js	6 hrs
Adding Frontend.js file in Project	4 hrs
Integrating Backend with Frontend	2 hrs

Table 7.9

List one thing that went particularly well this week

Adding Frontend.js file

List one thing that was the most challenging this week

Converting HTML into Pug.js

Self Evaluation: A

Week: 10

Description of Activity	Time Spent
Introduction to React.js and why we use them, Class-based and function-based components, Props.	4 hrs
States, Introduction to hooks in function-based components, Lifecycle methods like componentDidMount(), CSS module.	5 hrs
Form validation, Sending Https requests, Adding routes	4 hrs

Table 7.10

List one thing that went particularly well this week

Props and States

List one thing that was the most challenging this week

Hooks and Lifecycle methods

Self Evaluation: A

References

[1] https://stripe.com/docs

[2] https://www.npmjs.com/package/jsonwebtoken

[3] https://www.npmjs.com/package/bcryptjs

[4] https://www.npmjs.com/package/cookie-parser

[5] https://www.npmjs.com/package/express-mongo-sanitize

[6] https://www.npmjs.com/package/express-rate-limit

[7] https://www.npmjs.com/package/helmet

[8] https://en.wikipedia.org/wiki/Visual_Studio_Code

[9] https://en.wikipedia.org/wiki/Cmd.exe

[10] https://www.softwaretestinghelp.com/mongodb/robo-3t-robomongo-tutorial/

[11] https://creately.com/diagram-type/use-case

[12] https://en.wikipedia.org/wiki/Data-flow_diagram

[13] https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/

[14] https://www.smartdraw.com/entity-relationship-diagram/

[15] https://pugjs.org/api/getting-started.html

[16] https://docs.mongodb.com/stitch/mongodb/

[17] https://en.wikipedia.org/wiki/Express.js

[18] https://en.wikipedia.org/wiki/React_(JavaScript_library)

[19] https://mongoosejs.com/docs/documents.html

[20] https://en.wikipedia.org/wiki/Node.js

[21] https://www.npmjs.com/package/validator

[22] https://nodemailer.com/about/

[23] https://www.npmjs.com/package/random-token