THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA

FACULTY OF SCIENCE

A (LIVE) PROJECT REPORT

on

Food Shops, "The Food Adda"

for

Indian Innovation

Submitted by

Project No. 8

In fulfillment for the course of Semester-VI, named Mini Project (BCA1504)

of

BACHELOR OF COMPUTER APPLICATIONS

in

The Department of Computer Applications

January 19, 2020.

Guided by:

Ms. Khushbu Raval

Team members – Seat No:

Parth Vaswani – 523101
Aryaman Karde – 523038
Sharanam Chotai - 523022

Index

Project Completion Certificate	3
PREFACE	4
Acknowledgement	5
Requirements	9
Tools, Technologies & Libraries	10
Product Backlogs	13
Sprint Backlogs	14
Burndown Charts	17
Data Modelling	18
Data Dictionary	19
Screenshots & Explanations	21
Testing	45
Security and Other Features	54
Coding Standard	56
Limitations and Future Enhancements	57
References and Bibliography	58
Conclusion	59

Project Completion Certificate

M/S INDIAN INNOVATION

Paduali, Gola Kauriram Road, Kakrahi, Gorakhpur

Uttar Pradesh - 273408

Registered Number: 09BTGPR9623E1ZB

Date: 07/01/2021

Project Completion Letter

This is to Certify that Aryaman Karde, Parth Vaswani & Sharanam Chotai who are students of Third Year BCA, Maharaja Sayajirao University of Baroda, has successfully completed our firm's Web Application 'The Food Adda'. This work is original and has not been submitted anywhere else.

From:

M/S INDIAN INNOVATION

PREFACE

After completing 2 years in Department of Computer Applications successfully, we got to imply our knowledge of coding, teamwork, communication skills, tools, strategies into project, which is course of 2 credits.

"Mini Project", is a subject of 5th semester of BCA whose subject code is 'BCA1504', must be accomplished by every student through either dummy or live project in group of maximum 3 students.

"The Food Adda" is domain of group number 8 which involves (us) Parth, Aryaman & Sharanam and M/S Indian Innovation is our client for whom we have done it.

We are being guided by Ms. Khushbu Raval since starting phase of project.

As period of semester 5 is being completed, we are submitting this documentation to faculties of BCA which includes diagrams, strategies, dictionary, features and limitations of application.

In this document, we tried to make everything i.e., sections and diagrams with explanations, consistent, subsequently and easy to interpret by anyone who knows at least basics of software engineering.

As this document will be read by faculties as well as by few junior students of our succeeding batch (exemplar). With this concern in mind, we are putting strategies and methods along with appropriate graphics, which we have considered while working.

Acknowledgement

We take this opportunity to express our sincere gratitude:

To Head of the Department, Mr. Rakesh Srivastava for giving us opportunity to show our knowledge which we acquire in previous semesters, also for sharing your personal experiences of software development on-field.

To our passionate guide, Ms. Khushbu Raval for suggesting us throughout the project.

To our client who believed in our efforts and for his appraisals & calmness.

To all the faculties of Department of Computer Applications for putting efforts and managing situations even due to pandemic-lockdown. They showered their rain of knowledge on us when we had no idea about so many things and almost everything was new for us. Without them, we could not accomplish this job accurately.

About Project

In a brief, our project focuses on providing a platform for restaurants to go online, so that customers can avail the benefit of ordering food or reserving tables without visiting the store. It also provides Simple UI with beautiful UX design for shop owners to manage their shops and products. Last but not the least, we also have an admin panel for all the tasks that our admin is privileged for. All these comes as a SPA (Single Page Application).

Actual problem can be described as,

- We have provided an online approach to restaurants for reserving a table according to given time slot or book their food and take delivery.
    - Which means customer will reserve the table from website before he physically arrives. He needs to tell time of visiting, of course. Then he can choose whatever he/she wants from menu in traditional way. Customer will not have to wait till table gets empty. You can understand this by simple example of TOKEN. If you have token number or appointment you will get fast service.
    - In second scenario, customer desires to pack the food and take it to the home, so he can enjoy meal accompanied by his family. It was an example only. Anyways, he needs to select food items, to be packed in, and time, when he will arrive physically to take a delivery, through the website, specifically in this case.
- On other hand, shop owners should have access to manage orders, booked by customers. i.e., shop owner can confirm, reject, or fulfill the order through website merely. Nonetheless shop-owner will also manage their shops and products, update timings and availability through website.
    - Which means, owner of the shop needs to up to date his shop onto website whether shop is open or not, availability of products, and price of products.
    - He needs to continuously watch the new coming orders, which he can confirm or reject from the portal itself. In major cases, owner will accept the order, but exceptional phenomena happens where shop owner needs to deny for incoming orders.
    - As the customer will take the delivery of his order or will complete the meal, shop owner will mark it as fulfilled through website. Which can be seen by customer and shop owner both on the portal.
- The tasks of Administrator are:
    - To manage categories of food items,
        - Categories can be also said as classification of available food products.
        - For example, Italian, Indian, Punjabi, Chinese, etc. are food groups we added till now.

    - But it can be also be classified into dairy products, fruits, meat, vegetables, confections or sugary food, candy chocolate or cake.
    - Administrator will regulate them, by adding or removing categories inside of system.
- Blacklisting the inexcusable users of system,
    - There can be some users who is harmful to the firm, as their bad practices.
    - They need to be blacklisted from the system, which will be managed from administrator panel.
- Inspecting orders of every customer,
    - This will manage consistency of firm as it features upper level of organization/firm.
- Verifying newly added shop owners on the personal basis.
    - Any user can verify him through email, but for becoming a shop owner, verification from admin panel is mandatory step.

This was definition of our project.

This assignment was assigned to us. We are students of Final year of BCA and following table shows our contributions and participation throughout whole development:

Name	Lines of Code (LOC)	Time (in hour)
Parth Vaswani	6949	88
Aryaman Karde	4310	41
Sharanam Chotai	4235	44
Total	**15494**	**173**

For accomplishment of the assigned work, we used Agile Methodology.

Agile is an umbrella term for a set of frameworks and practices. It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages flexible responses to change.

One of the differences between agile software development methods and traditional methods is the approach to quality and testing. In the traditional method, work moves through Software Development Lifecycle (SDLC) phases—with one phase being completed before another can start—hence the testing phase is separate and follows a build phase. However, in agile software development testing is completed in the same iteration (sprint) as programming.

Because testing is done in every iteration—which develops a small piece of the software—users can frequently use those new pieces of software and validate the value. After the users know the real value of the updated piece of software, they can make better decisions about the software's future. Having a value retrospective and software re-planning session in each iteration—Scrum typically has iterations of just two weeks which is also known as sprint—helps the team continuously adapt its plans to maximize the value it delivers. This follows a pattern like the Plan-Do-Check-Act (PDCA) cycle, as the work is planned, done, checked (in the review and retrospective), and any changes agreed are acted upon.

This iterative approach supports a product rather than a project mindset. This provides greater flexibility throughout the development process; whereas on projects the requirements are defined and locked down from the very beginning, making it difficult to change them later. Iterative product development allows the software to evolve in response to changes in business environment or market requirements.

In agile, we were using scrum framework specifically (in which sprints take place). Every sprint is of 2 weeks, you can see more about agile in project in subsequent section.

Requirements

For server (Hardware):

- ✓ CPU(s): x86-64 (Intel is preferred); 4 cores minimum (16 or more recommended).
- ✓ Low-voltage CPUs (i.e., Intel Atom) are strongly not recommended due to poor performance.
- ✓ RAM: 8GB minimum, 16GB or more recommended.
- ✓ Disk: 64 GB or more of usable storage for one-server installation, including:
  - 30 GB minimum for OS,
  - 35 GB for the infrastructure,

For client:

- ✓ Any of the following Operating System:
  - Android 5 and newer
  - iOS 10 and newer
  - Microsoft Windows 7 or higher
  - Any Linux distribution (with recommended web browser)

- ✓ Any of the following Web Browser (latest stable version recommended):
  - Chrome Browser
  - Android Browser
  - Edge Browser
  - Internet Explorer Browser (from version 11)
  - Firefox Browser
  - Opera Browser
  - Safari Browser
  - iPhone/iPad Browser (from version 6)

Tools, Technologies & Libraries

This application is built using tools and upon technologies as follows:

- Front end:
  - React
  - Libraries:
    - Dev dependencies
      - eslint
      - eslint-config-airbnb
      - eslint-config-prettier
      - eslint-plugin-import
      - eslint-plugin-jsx-a11y
      - eslint-plugin-react
      - eslint-plugin-react-hooks
    - Dependencies
      - @material-ui/core // built-in designed components
      - @material-ui/icons // vector icons
      - @material-ui/lab // advance components
      - axios // async http request, CRUD operations
      - easy-peasy // quick and easily manages state
      - eslint-config-airbnb
      - eslint-config-prettier
      - jwt-decode // decode jwt tokens
      - react // primary technology
      - react-dom // manages dom element of webpage
      - react-router-dom // navigate inside components, for URL
      - react-scripts
      - workbox-cacheable-response // for caching responses
      - workbox-core // built to be modular
      - workbox-expiration // limit no. of entries in a cache
      - workbox-precaching // updates assets in cache.
      - workbox-routing // route requests to different functions.
      - workbox-strategies // implementation of strategies

- Back end:
  - Mongo DB // NoSQL Database
  - Express JS // Back end framework
  - NodeJS // JavaScript runtime
  - Libraries:
    - Dev dependencies
      - concurrently
      - eslint
      - eslint-config-airbnb-base
      - eslint-config-prettier
      - eslint-plugin-import
      - nodemon
    - Dependencies
      - bad-words
      - bcryptjs
      - body-parser
      - compression
      - cors
      - express
      - express-fileupload
      - express-mongo-sanitize
      - express-rate-limit
      - helmet
      - hpp
      - jsonwebtoken
      - mongoose
      - morgan
      - nodemailer
      - passport
      - passport-jwt
      - rate-limit-mongo
      - sharp
      - spdy
      - validator
      - xss-clean

- Tools & Platforms:

Visual Studio Code	// Code Editor
Postman	// For API Testing & Exclusive API-Documentation
Heroku	// Backend deployment
Netlify	// Frontend deployment
MongoDB Atlas	// Deployment of Database
Git / Github	// Version Control System
Chrome Dev Tools	// For debugging

Product Backlogs

A product backlog is a list of the new features, changes to existing features, and other activities that a team may deliver to achieve a specific outcome. The product backlog is the single authoritative source for things that a team works on.

Priority	Item	Description	Est	By	Sprint
Very High					
	1	Create boilerplate for project (Back end)	4	PV	1
	2	Create boilerplate for project(Front end)	2	A&S	1
	3	Database relations and data dictionary	8	S	1
	4	Database	5	PV	1
	5	Security	6	PV	1
		XSS(cross site scripting)			
		Regex DOS			
		DOS			
		NoSQL injection			
		hpp(http parameter pollution)			
High					
	6	Authentication and authorization (Backend)	6	PV	2
		Log in			
		Sign up			
		Delete			
	7	Authentication and authorization (Front end)	8	A&S	2
		Log in			
		Sign up			
		Delete			
	8	Manage Shops (Backend)	4	PV	2
		Shop			
		Image			
	9	Manage Shops (Front end)	4	A&S	3
		Shop			
		Image			
	10	Manage Products(Backend)	4	PV	2
		Product			
		Image			
	11	Manage Products(Front end)	4	A&S	3
		Product			
		Image			
Medium					
	12	Search Bar for searching shops and products (Customer)(Backend)	2	PV	3
		Product Search			
		Shop Search			
	13	Admin Actions(Backend)	2	PV	3
		Category			
		Verification			
		Delete			
	14	API Documentation (Backend)	16	PV	4
	15	Homepage	8	A&S	3
	16	Manage orders (Customer)(Backend)	5	PV	3
		Products			
		Order Status			
		Order			
	17	Manage reviews (Customer)(Backend)	2	PV	3
		Review			
		Voting			
	18	Manage bookmarks (Customer)(Backend)	1	PV	3
	19	Search Bar for searching shops and products (Frontend)	2	A&S	3
	20	Manage Orders (Shop Owners)(Backend)	1	PV	3
		Order Status			
	21	Manage Orders (Customer) (Front end)	8	A&S	5
		Products			
		Order Status			
		Order			
	22	Manage Orders (Shop Owner) (Front end)	2	A&S	5
		Order Status			
	23	Manage Reviews (Front end)	4	A&S	5
		Review			
		Voting			
	24	Manage Bookmarks (Front end)	2	A&S	5
		Bookmarks			
	25	Admin Actions (Front end)	4	A&S	5
		Category			
		Verification			
		Delete			
Low					
	26	PWA	1	PV	2
	27	Alpha testing (Back-end)	16	PV	4
	28	Alpha testing (Front-end)	16	A&S	4
	29	Deployment	1	PV	4

Sprint Backlogs

It represents the primary output of sprint planning. In our project, every sprint was scheduled 2 weeks in advance.

				Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	
Sprint #1 Backlog	**Assigned to**	**Status**	**Original Estimated hours**	**13-Oct**	**14-Oct**	**15-Oct**	**16-Oct**	**17-Oct**	**18-Oct**	**19-Oct**	**20-Oct**	**21-Oct**	**22-Oct**	**23-Oct**	**24-Oct**	**25-Oct**	**26-Oct**	**27-Oct**	**Sprint Review**
Backend																			
Create boilerplate for project																			
Create folder structure	Parth Vaswani	Completed	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create package.json	Parth Vaswani	Completed	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Add dependencies	Parth Vaswani	Completed	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Setup boilerplate code	Parth Vaswani	Completed	0.5	0.5	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Setup config files	Parth Vaswani	Completed	1	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Setup git	Parth Vaswani	Completed	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Setup git hub repository	Parth Vaswani	Completed	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Setup eslint	Parth Vaswani	Completed	0.3	0.3	0.3	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0
Setup prettier	Parth Vaswani	Completed	0.2	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create license	Parth Vaswani	Completed	0.3	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Setup development and production scripts.	Parth Vaswani	Completed	1	1.5	1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Database relations and data dictionary	Sharanam Chotai	Completed	9	8.5	8.5	8	8.6	8.6	7	5.5	4	2	1	0	0	0	0	0	0
Database																			
Validate data before saving it in database	Parth Vaswani	Completed	3	3	3	3	3	3	3	2.8	2.5	2	2.3	1.6	1.6	1	0	0	0
Create database models	Parth Vaswani	Completed	3	3	3	3	3	3	3	3	2.5	2.5	1	1.5	1.5	1	0	0	0
Security																			
XSS(cross site scripting)																			
Add validation for user generated data	Parth Vaswani	Completed	1	1	0.5	0.8	0	0	0.2	0	0	0	0	0	0	0	0	0	0
Add xss-clean as a middleware	Parth Vaswani	Completed	1	1	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.2	0	0	0
Add helmet as a middleware	Parth Vaswani	Completed	1	1	1	1	1	1	1	1	1	1	1	1	1	0.5	0.5	0	0
Regex DOS																			
Clean and escape regular expressions before using them	Parth Vaswani	Completed	1	1	1	1	1	1	0.8	0.8	0.5	0	0	0	0	0.5	0	0	0
DOS																			
Add express-mongo-sanitize as a middleware	Parth Vaswani	Completed	1	1	1	1	1	1	1	0.5	0	0	0	0	0	0	0	0	0
Setup rate-limit-mongo	Parth Vaswani	Completed	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.5	0.5	0
NoSQL injection																			
Add express-mongo-sanitize as a middleware	Parth Vaswani	Completed	1	1	1	1	1	1	1	1	0.5	0	0	0	0	0	0	0	0
hpp(http parameter pollution)																			
Add hpp as a middleware	Parth Vaswani	Completed	1	2	2	2	2	2	2	2	2	1.5	1.5	1.5	1.5	1	0.5	0.5	0
Frontend																			
Create boilerplate for project																			
Create folder structure	Aryaman Karde	Completed	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create package.json	Aryaman Karde	Completed	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Add dependencies	Aryaman Karde	Completed	0.2	0.2	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0
Setup boilerplate code	Aryaman Karde	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.2	0.3	0.2	0	0	0	0	0	0	0	0
Setup eslint	Aryaman Karde	Completed	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0	0	0	0	0	0
Setup prettier	Aryaman Karde	Completed	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0	0	0	0	0	0
Total			29.5	30.5	27	23.8	23.4	23.4	21.7	19.2	15.5	11.3	9.1	7.4	7.4	5.2	1.5	1	0

				Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	
Sprint #2 Backlog	**Assigned to**	**Status**	**Original Estimated hours**	**29-Oct**	**30-Oct**	**31-Oct**	**01-Nov**	**02-Nov**	**03-Nov**	**04-Nov**	**05-Nov**	**06-Nov**	**07-Nov**	**08-Nov**	**09-Nov**	**10-Nov**	**11-Nov**	**Sprint Review**
Backend																		
Authentication and authorization																		
Log in																		
Customer login with email and password	Parth Vaswani	Completed	0.6	0.6	0.3	0.3	0.2	0	0	0	0	0	0	0	0	0	0	0
Shop owner login with password and email	Parth Vaswani	Completed	0.6	0.6	0.3	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0
Admin login with password, email and advanced password	Parth Vaswani	Completed	0.6	0.6	0.6	0.6	0.4	0	0	0	0	0	0	0	0	0	0	0
Resend email for forget password	Parth Vaswani	Completed	0.6	0.6	0.6	0.6	0.4	0.3	0	0	0	0	0	0	0	0	0	0
Forget Password	Parth Vaswani	Completed	0.6	0.6	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0
Save hashed password	Parth Vaswani	Completed	0.3	0.3	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0
Sign up																		
Add details for sign up and register	Parth Vaswani	Completed	0.6	0.6	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0
Verify email	Parth Vaswani	Completed	0.5	0.5	0.5	0.5	0.3	0.1	0	0	0	0	0	0	0	0	0	0
Resend email for verification	Parth Vaswani	Completed	0.6	0.6	0.6	0.6	0.3	0.1	0	0	0	0	0	0	0	0	0	0
Delete																		
Delete account except admin	Parth Vaswani	Completed	0.5	0.5	0.5	0.5	0.5	0.3	0.3	0.1	0.1	0	0	0	0	0	0	0
On deletion of account delete all related shops, products and reviews	Parth Vaswani	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.3	0.1	0.3	0	0	0	0	0	0	0
PWA																		
Functionality																		
Add install prompt	Parth Vaswani	Completed	0.5	0.5	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0
Add icons	Parth Vaswani	Completed	0.2	0.2	0.2	0.2	0	0.2	0	0	0	0	0	0	0	0	0	0
Add caching for offline use	Parth Vaswani	Completed	0.3	0.3	0.3	0.3	0.3	0.1	0	0	0	0	0	0	0	0	0	0
Manage Shops																		
Shop																		
Create a shop	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.2	0	0	0	0	0	0	0	0	0	0	0
Edit a shop	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.2	0	0	0	0	0	0	0	0	0	0	0
Delete a shop	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0	0	0	0	0	0	0	0	0
Delete all the shops at a time	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0.2	0.2	0	0	0.1	0.1	0	0	0
Change Shop timing	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0.2	0.2	0	0	0.1	0.1	0	0	0
Change shop's availability status	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.3	0.1	0.2	0	0	0	0.1	0	0	0	0
Review shop	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0	0.2	0	0	0	0	0	0	0
Image																		
Add image to the shop	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.3	0.1	0.2	0	0	0	0	0	0	0	0
Remove image of shop	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.2	0	0	0	0	0	0	0	0
Validate, compress and resize images	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.3	0.2	0	0	0	0	0	0	0
Manage Products																		
Product																		
Create a Product	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.2	0	0	0	0	0	0	0	0	0	0	0
Edit a product	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.2	0	0	0	0	0	0	0	0	0	0	0
Delete a product	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0	0	0	0	0	0	0	0	0
Delete all the products of a particular shop	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0.2	0.2	0	0	0.1	0.1	0	0	0
Change product's availability status	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0.2	0.2	0	0	0.1	0.1	0	0	0
Review product	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.5	0.3	0	0.2	0	0	0	0	0	0	0
Image																		
Add image to the product	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0	0	0	0
Remove image of a product	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.4	0	0.1	0
Validate, compress and resize image	Parth Vaswani	Completed	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.1	0.2	0	0
Frontend																		
Authentication and authorization																		
Log in																		
Customer login with email and password	Aryaman, Sharanam	Completed	0.8	0.8	0.6	0.6	0.4	0.2	0	0	0	0	0	0	0.2	0	0	0
Shop owner login with password and email	Aryaman, Sharanam	Completed	0.8	0.8	0.7	0.7	0.4	0.2	0	0	0	0	0	0.2	0	0	0	0
Admin login with password, email and advanced password	Aryaman, Sharanam	Completed	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Resend email for forget password	Aryaman, Sharanam	Completed	0.8	0.8	0.5	0.5	0.5	0	0	0	0	0	0	0.2	0	0	0	0
Forget Password	Aryaman, Sharanam	Completed	0.8	0.8	0.8	0.8	0.6	0.7	0.4	0.4	0.4	0.3	0.3	0	0.2	0	0.1	0
Sign up																		
Add details for sign up and register	Aryaman, Sharanam	Completed	0.8	0.8	0.8	0.8	0.6	0.7	0.4	0.2	0.2	0	0	0.3	0.3	0	0	0
Verify email	Aryaman, Sharanam	Completed	0.8	0.8	0.8	0.8	0.6	0.7	0.5	0.2	0	0	0	0.2	0	0	0.1	0
Resend email for verification	Aryaman, Sharanam	Completed	0.8	0.8	0.8	0.8	0.6	0.7	0.4	0.2	0	0	0	0	0	0	0	0
Delete																		
Delete account except admin	Aryaman, Sharanam	Completed	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
On deletion of account delete all related shops, products and reviews	Aryaman, Sharanam	Completed	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Total			22.8	22.8	20.4	20.4	16.6	13.3	8.7	6.7	6.2	4.1	4.1	4.8	4.2	2.8	2.9	2.6

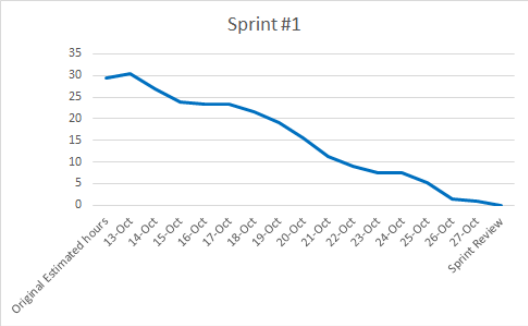
				Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	
Sprint #3 Backlog	Assigned to	Status	Original Estimated hours	13-Nov	14-Nov	15-Nov	16-Nov	17-Nov	18-Nov	19-Nov	20-Nov	21-Nov	22-Nov	23-Nov	24-Nov	25-Nov	26-Nov	27-Nov	Sprint Review
Backend																			
Search Bar for searching shops and products (Customer)																			
Product Search																			
Vallidate query params	P	Completed	0.2	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Enable filter search	P	Completed	0.2	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Enable sorting	P	Completed	0.2	0.2	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0
Paginate results	P	Completed	0.2	0.2	0.2	0.2	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0
Shop Search																			
Vallidate query params	P	Completed	0.2	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Enable filter search	P	Completed	0.2	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Enable sorting	P	Completed	0.2	0.2	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0
Paginate results	P	Completed	0.2	0.2	0.2	0.2	0.2	0.2	0	0	0.2	0.1	0.1	0	0	0	0	0	0
Manage orders (Customer)																			
Products																			
Add product	P	Completed	0.5	0.2	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0
Remove product	P	Completed	0.5	0.5	0.4	0.4	0	0.6	0	0	0	0	0	0	0	0	0	0	0
Change quantity of product	P	Completed	0.5	0.5	0.5	0.5	0.3	0.2	0	0	0	0	0	0	0	0	0	0	0
Order Status																			
Close order	P	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0	0	0.6	0	0	0	0	0	0
Cancel order	P	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.2	0	0	0	0	0	0	0	0	0
Finalize order	P	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.3	0.2	0.1	0	0	0	0	0	0	0
Order																			
Create order	P	Completed	0.5	0.5	0.2	0.2	0	0	0.5	0.4	0.1	0	0	0	0	0	0	0	0
Edit order	P	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.2	0.3	0.5	0	0	0	0	0
Check if the time slot, product and shop is available	P	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.2	0.3	0.5	0	0	0	0	0
Review order	P	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.2	0.3	0.5	0	0	0	0	0
Manage reviews (Customer)																			
Review																			
Add review	P	Completed	0.3	0.2	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
Edit review	P	Completed	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0	0	0	0	0.1	0	0	0	0	0
Filter reviews	P	Completed	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0	0	0	0	0	0	0	0	0
Get reviews	P	Completed	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0	0	0	0	0	0	0	0	0
Filter curse words from reviews	P	Completed	0.3	0.3	0.3	0.3	0.2	0.3	0.3	0.2	0	0	0	0	0	0	0	0	0
Voting																			
Add vote	P	Completed	0.3	0.2	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
Delete vote	P	Completed	0.3	0.3	0.3	0.3	0.3	0.2	0	0	0	0	0	0	0	0	0	0	0
Manage bookmarks (Customer)																			
Bookmarks																			
Add bookmark	P	Completed	0.3	0.2	0.2	0.2	0.1	0	0.4	0	0	0	0	0	0	0	0	0	0
Remove bookmark	P	Completed	0.3	0.3	0.3	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0
Review own bookmark	P	Completed	0.3	0.3	0.3	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0
Manage Orders (Shop Owners)																			
Order Status																			
Confirm order	P	Completed	0.3	0.3	0.3	0.3	0.2	0	0	0	0	0	0	0	0	0	0	0	0
Reject order	P	Completed	0.3	0.3	0.3	0.3	0.3	0.1	0	0	0	0	0	0	0	0	0	0	0
Fulfill order	P	Completed	0.3	0.3	0.3	0.3	0.3	0.1	0.2	0	0	0	0	0	0	0	0	0	0
Review orders	P	Completed	0.3	0.3	0.3	0.3	0.3	0.1	0	0	0	0	0	0	0	0	0	0	0
Admin Actions																			
Category																			
Add product category	P	Completed	0.3	0.2	0	0	0	0	0.5	0.4	0.3	0.2	0.2	0	0	0	0	0	0
Remove category	P	Completed	0.3	0.3	0.3	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0
Get categories	P	Completed	0.3	0.3	0.3	0.3	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0
Verification																			
Verify shop owner	P	Completed	0.3	0.3	0.3	0.3	0.3	0.3	0.1	0	0	0	0	0	0	0	0	0	0
Add user to blacklist	P	Completed	0.3	0.3	0.3	0.3	0.1	0	0	0.4	0	0	0	0	0	0	0	0	0
Remove user from blacklist	P	Completed	0.3	0.3	0.3	0.3	0.1	0	0	0.4	0	0	0	0	0	0	0	0	0
Delete																			
Delete all shops of an owner	P	Completed	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.1	0.1	0	0	0	0	0	0
Frontend																			
Homepage																			
Prepare Landing Page	AS	Completed	8	8	8	8	7.5	7	7.2	6.5	6.1	5.7	5.7	4.3	4	2.8	1.1	0.7	0
Search Bar for searching shops and products																			
Dropdown to select category of search																			
Search for shop	AS	Completed	0.5	0.3	0.3	0.3	0.1	0	0.1	0	0	0	0	0	0	0	0	0	0
Search for product	AS	Completed	0.5	0.5	0.5	0.5	0.3	0.2	0.3	0.1	0	0	0	0	0	0	0	0	0
Search bar																			
Input box with list of suggestions	AS	Completed	0.5	0.5	0.5	0.5	0.5	0.3	0.1	0	0	0	0	0	0	0	0	0	0
Authentication and authorization																			
Log In																			
Admin login with password, email and advanced password	AS	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.3	0.2	0.2	0	0	0	0	0	0
Delete																			
Delete account except admin	AS	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.3	0.1	0	0.3	0.1	0
On deletion of account delete all related shops, products and reviews	AS	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.2	0.1	0	0	0	0
Total			23.9	22.6	21.5	21.5	18.2	15.8	15.2	12.7	9.7	8	8.8	6.4	4.2	2.8	1.4	0.8	0

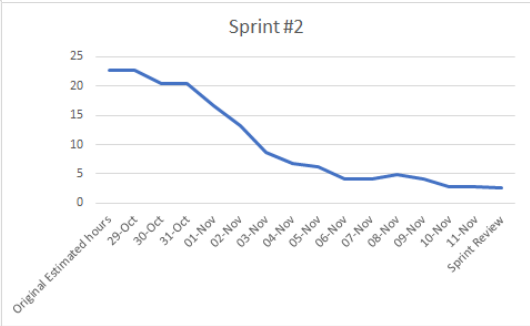
				Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	
Sprint #4 Backlog	Assigned to	Status	Original Estimated hours	05-Dec	06-Dec	07-Dec	08-Dec	09-Dec	10-Dec	11-Dec	12-Dec	13-Dec	14-Dec	15-Dec	16-Dec	17-Dec	18-Dec	19-Dec	Sprint Review
Backend																			
API documentation																			
Docs																			
Setup collection in postman	P	Completed	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create all endpoints	P	Completed	9	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Add description to endpoints	P	Completed	6	6	6	5	3	2	2.5	1.5	0.5	0	0	0	0	0	0	0	0
Frontend																			
Manage Shops(Shop Owner)																			
Shop																			
Create a shop	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0	0
Edit a shop	A	Completed	0.5	0.5	0.5	0.5	0.5	0	0.3	0	0	0	0	0	0	0	0	0	0
Delete a shop	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0
Delete all the shops at a time	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0
Change Shop timing	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0
Change shop's availability status	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0
Review shop	A	Completed	0.5	0.5	0.5	0.5	0.5	0	0.3	0	0	0	0	0	0	0	0	0	0
Image																			
Add image to the shop	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0	0
Remove image of shop	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0
Manage Products (Shop Owner)																			
Product																			
Create a Product	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0.2	0	0	0
Edit a product	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0.3	0	0	0	0
Delete a product	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0
Delete all the products of a particular shop	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0.3	0	0	0	0
Change product's availability status	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.3	0	0	0	0	0
Review product	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.2	0	0	0
Image																			
Add image to the product	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0	0
Remove image of a product	A	Completed	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0
Shops and Product																			
Pages																			
User Interface	S	Completed	2	1.5	1.4	1.5	1	0.5	1	1.2	0.5	0.3	0	0	0	0.2	0	0	0
Page Layout	S	Completed	2	2	0.5	0	0.5	0.5	0.5	0.5	0.5	0	0.3	0.5	0.3	0	0	0	0
Add Dynamic Route Imports	P	Completed	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create Component Folder Structure and Split	P	Completed	8	6	6	6	6	6	6	6	6	6	6	5	3	4	2	1.5	0
Add Meta Tags	P	Completed	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0	0.5	0	0
Deployment and Testing																			
Deployment																			
Add deployment Scripts	P	Completed	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Setup Environment Variables	P	Completed	1	1	1	1	1	1	0	0	0	0.5	1	0.3	0	0	0	0	0
Setup github to heroku pipeline	P	Completed	1	1	1	1	1	1	1	0.5	0	0.5	1	0.3	0	0	0	0	0
Setup database at mongoDB atlas	P	Completed	1	1	1	1	1	1	1	1	0.5	0.5	0	0	0.3	0.3	0	0	0
	Total		41.5	37.5	30.9	25.5	23.5	20.5	20.1	16.7	12.5	10.3	10.8	6.9	4.7	4.9	2.5	1.5	0

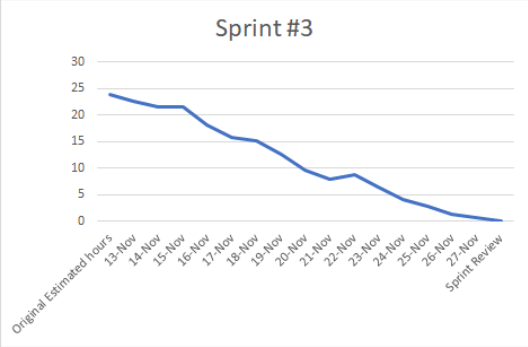
				Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	
Sprint #5 Backlog	Assigned to	Status	Original Estimated hours	21-Dec	22-Dec	23-Dec	24-Dec	25-Dec	26-Dec	27-Dec	28-Dec	29-Dec	30-Dec	31-Dec	01-Jan	02-Jan	03-Jan	04-Jan	05-Jan	06-Jan	Sprint Review
Backend																					
Alpha Testing	P	Completed	16	16	15.5	14	14.5	13	12	13.8	11.5	11.5	11.5	11.5	10	8	7.5	5	3	1.5	0
Frontend																					
Manage Orders (Customer)																					
Products																					
Add Product	AS	Completed	1	1	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Remove Product	AS	Completed	1	1	1	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Change quantity of product	AS	Completed	1	1	1	1	1	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0
Order Status																					
Close Order	AS	Completed	1	1	1	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Finalize Order	AS	Completed	1	1	1	1	1	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0
Cancel Order	AS	Completed	1	1	1	1	1	0.5	0	0.5	0.5	0	0	0	0	0	0	0	0	0	0
Order																					
Create Order	AS	Completed	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Edit Order	AS	Completed	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Manage Orders (Shop Owner)																					
Order Status																					
Confirm Order	AS	Completed	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reject Order	AS	Completed	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Review Order	AS	Completed	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fulfill Order	As	Completed	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Manage Reviews																					
Review																					
Add Review	AS	Completed	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.5	0.3	0.5	0	0	0
Edit Review	AS	Completed	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.7	0.5	0.6	0.2	0	0
Filter Reviews	AS	Completed	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.3	0.5	0	0	0	0
Voting																					
Add Vote	AS	Completed	0.8	0.8	0.8	0.8	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Remove Vote	AS	Completed	0.8	0.8	0.8	0.8	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Manage Bookmarks																					
Bookmarks																					
Add bookmark	AS	Completed	0.7	0.7	0.7	0.7	0.7	0.7	0.3	0	0	0	0	0	0	0	0	0	0	0	0
Remove bookmark	AS	Completed	0.7	0.7	0.7	0.7	0.7	0.7	0.3	0	0	0	0	0	0	0	0	0	0	0	0
Review own bookmarks	AS	Completed	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.5	0.3	0	0	0	0
Alpha Testing	AS	Completed	16	16	16	15	15.5	14	14	13	12.5	12.5	12.5	12.5	11	10	7.5	5	4.5	3	0
Admin Actions																					
Category																					
Add Product Category	AS	Completed	0.7	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Remove Product Category	AS	Completed	0.7	0.7	0.7	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Verification																					
Verify shop owners	AS	Completed	0.7	0.5	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Add user into blacklist	AS	Completed	0.7	0.7	0.5	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Remove user from blacklist	AS	Completed	0.7	0.7	0.3	0.5	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Delete																					
Delete all shops of an owner	AS	Completed	0.7	0.7	0.7	0.7	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Total		52.3	50.9	47.5	43.3	39.9	34	30.7	30.4	27.6	27.1	27.1	27.1	24.1	20	16.6	11.1	7.7	4.5	0

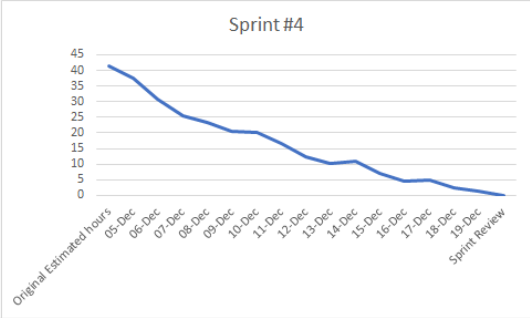
Burndown Charts

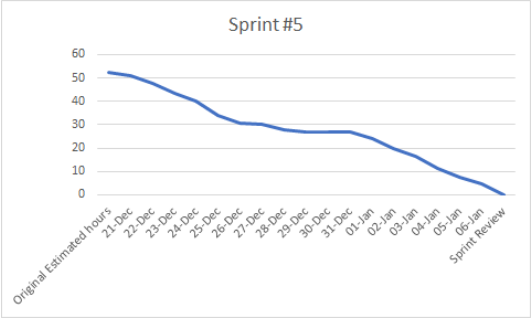
It is a graphic representation of how quickly the team is working through backlog items. It is used to capture a description of a feature from an end-user perspective. It shows the total effort against the amount of work for each sprint.







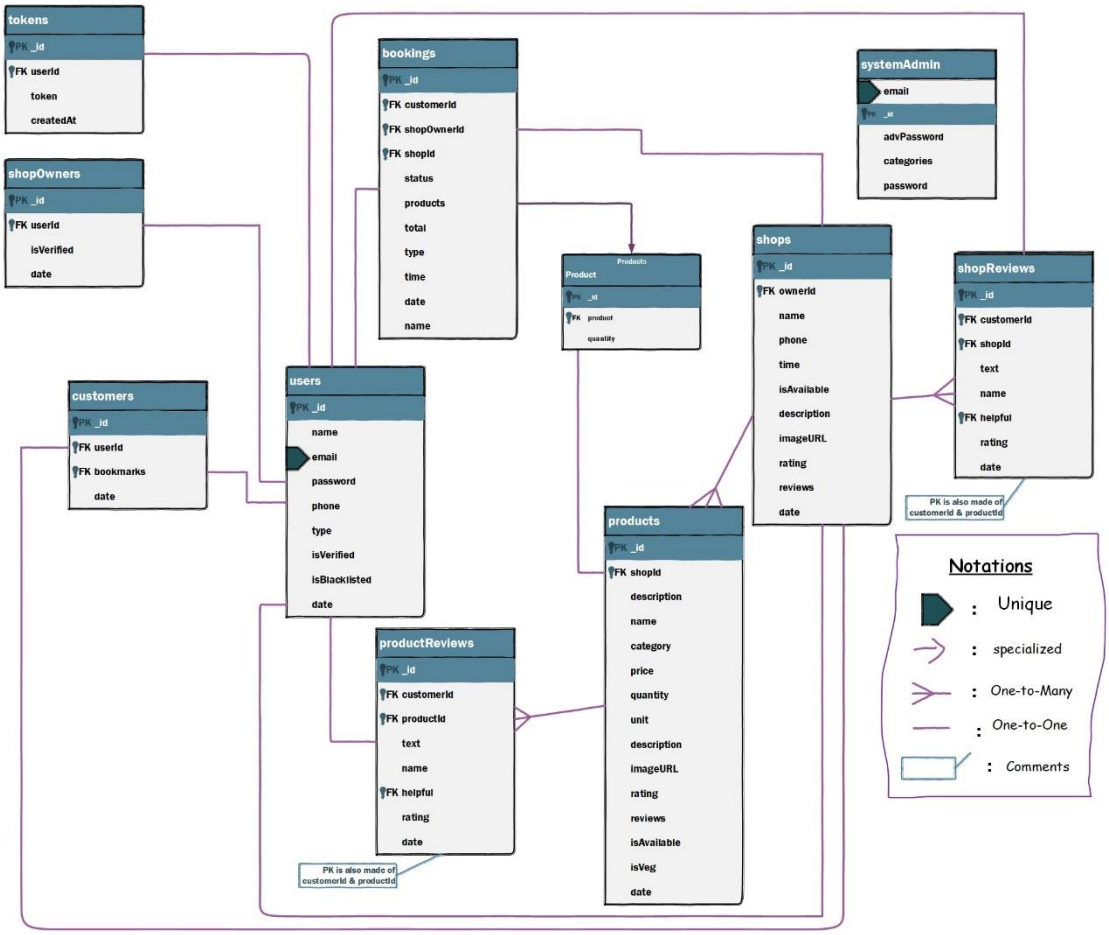




Data Modelling

Following diagram describes schema, which is implemented in MongoDB. MongoDB is a cross-platform document-oriented database program and classified as a NoSQL database program. It uses JSON-like documents with optional schemas.

Instead of entity-relationship diagram crow's foot notations are used here for ease.



Data Dictionary

It is a tabular description of database schema, by which developers can imply database.

Products Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
shopId	ObjectID	shops	TRUE				Shop Id is foreign key from 'shops', type is objectID of schema, required.
name	String		TRUE				Name is string, required.
category	String		TRUE				Category is string, required.
price	Number		TRUE				Price is number, required.
quantity	Number		TRUE			gt 1	Quantity is number, required, greater than 1.
unit	String		TRUE				Unit is string, required
description	String		TRUE			20 < x < 500	Description is required string, Length must be between 20 and 500.
imageURL	String						Image URL is string.
rating	Number				0		Rating is a number.
reviews	Number				0		Reviews is number, default value is 0
isAvailable	Boolean				FALSE		Isavailable is boolean, default value is FALSE
isVeg	Boolean				FALSE		Isveg is boolean, default value is FALSE
date	Date				Current Date		Date is date, default value is Current Date

Shops Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
name	String		TRUE				name is string, required.
ownerId	ObjectID	users	TRUE				ownerId is objectid from 'users', required.
phone	String		TRUE				phone is string, required.
isAvailable	Boolean				FALSE		isAvailable is boolean, required, by default FALSE
time	String		TRUE				time is string, required.
description	String		TRUE			20 < x < 500	description is string, required, must be between 20 and 500.
imageURL	String						imageURL is string, required.
rating	Number				0		rating is number, required, 0 as initial state.
reviews	Number				0		reviews is number, required, 0 as initial state.
date	Date				Current Date		date is type of date, required, Current date Default.

Booking Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
customerId	ObjectID	users	TRUE				Customer Id is foreign key from 'users', type is objectID of schema, required.
shopOwnerId	ObjectID	users	TRUE				Shop Owner Id is foreign key from 'users', type is objectID of schema, required.
shopId	ObjectID	shops	TRUE				Shop ID is foreign key from 'shops', type is objectID of schema, required.
status	String				selecting	enum	Status contains any one of the given strings in enumeration. Default or initial state is 'Selecting'.
products	Object[ ]					Array of JS-Object	Products contains array of object, which has properties : product and quantity (specified in next table)
total	Number				0	GreaterThan 0	Type of total is Number. It must not be 0 or less Rupees.
type	String		TRUE			enum	Type contains book&eat or order&pick, as given in enum.
time	String		TRUE				Time is stored as string in DB and is required object.
date	Date				Current Date		Date is required to be stored in DB. Current date should be automatically logged in.

Product Review Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
customerId	ObjectId	users					customerID is foreign key from 'users'
productId	ObjectId	products					productID is foreign key from 'products'
text	String		TRUE			8 < x < 500	text is String, required and length must be between 8 and 500
name	String		TRUE				name is string, required.
helpful	ObjectId[ ]	users					helpful is array of objectid, from users schema.
rating	Number		TRUE			0.5 < x < 5	rating is Number, required and length must be between 0.5 and 5
date	Date				Current Date		date is date, default current date.

Shop Reviews Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
customerId	ObjectId	users					customerID is foreign key from 'users'
productId	ObjectId	products					productID is foreign key from 'products'
text	String		TRUE			8 < x < 500	text is String, required and length must be between 8 and 500
name	String		TRUE				name is string, required.
helpful	ObjectId[ ]	users					helpful is array of objectid, from users schema.
rating	Number		TRUE			0.5 < x < 5	rating is Number, required and length must be between 0.5 and 5
date	Date				Current Date		date is type of date, required, Current date Default.

Users Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
name	String		TRUE				name is type of String, required.
email	String		TRUE	TRUE		lowercase	email is type of String, required, unique, must be in lowercase.
password	String		TRUE				password is type of String, required.
phone	String		TRUE				phone is type of String, required, unique.
type	String				customer	enum	Type of user is String, default value Customer, shop_owner is another type of user.
isVerified	Boolean				FALSE		it is to check weather user is verified or not.
isBlacklisted	Boolean				FALSE		it is to check weather user is blacklisted or not.
date	Date				Current Date		date is type of date, required, Current date Default.

Admin Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
email	String		TRUE	TRUE		lowercase	email is a required, unique, string object, which must be in lower-case.
password	String		TRUE				password is required, string object
advPassword	String		TRUE				advance password is required, string object
categories	String[ ]		TRUE				it is array of category and category is required, string object

Customer Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
userId	ObjectID	users	TRUE				User Id is foreign key from 'users', type is objectID of schema, required.
bookmarks	ObjectID[ ]	shops	TRUE				Type of bookmarks is array of objectID of schema 'users'. It is required.
date	Date				Current Date		Date is required to be stored in DB. Current date should be automatically logged in.

Tokens Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
userId	ObjectID	users	TRUE				User Id is foreign key from 'users', type is objectID of schema, required.
token	String		TRUE				Token is String, required.
createdAt	Date		TRUE		Current Date	expires in hour	Creation time is required. Token will be expire in 1 hour (3600 seconds)

'Products' in Booking Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
product	Object	Project					Products Object
quantity	Number						Quantity of specified product is number.

Shop Owner Schema

Object/ Column	Type	Reference	Required	Unique	Default	Remarks	Read it as
userId	ObjectId	users	TRUE				User Id is foreign key from 'users', type is objectID of schema, required.
isVerified	Boolean				FALSE		it is boolean and initially false
date	Date				Current Date		date is type of date, required, Current date Default.

Screenshots & Explanations

Visual Studio Code was our only code editor and development environment (IDE).

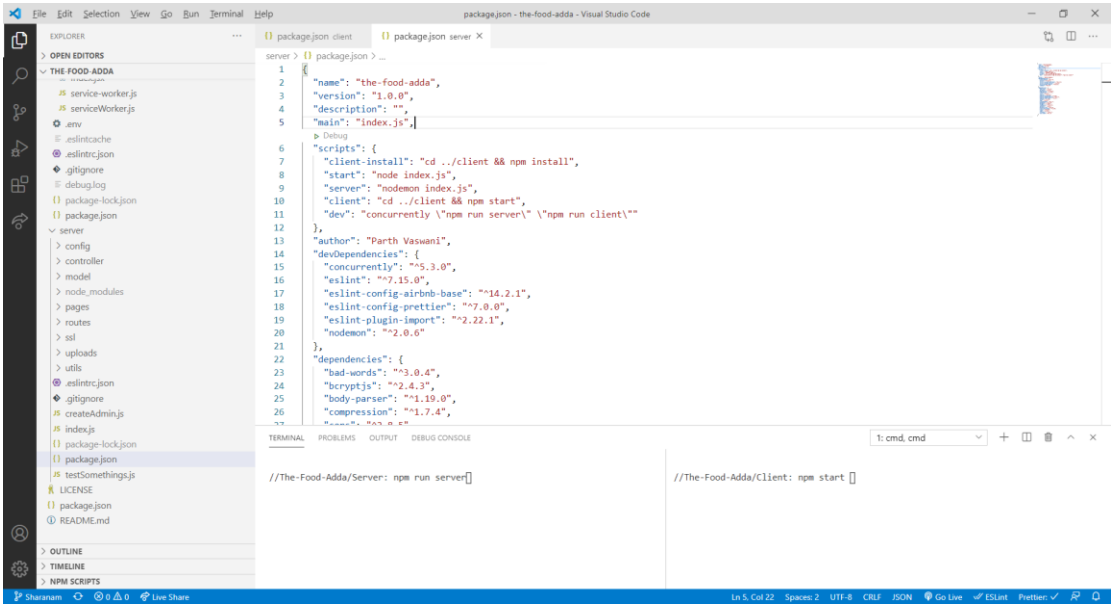


Fig: Visual studio code on desk.

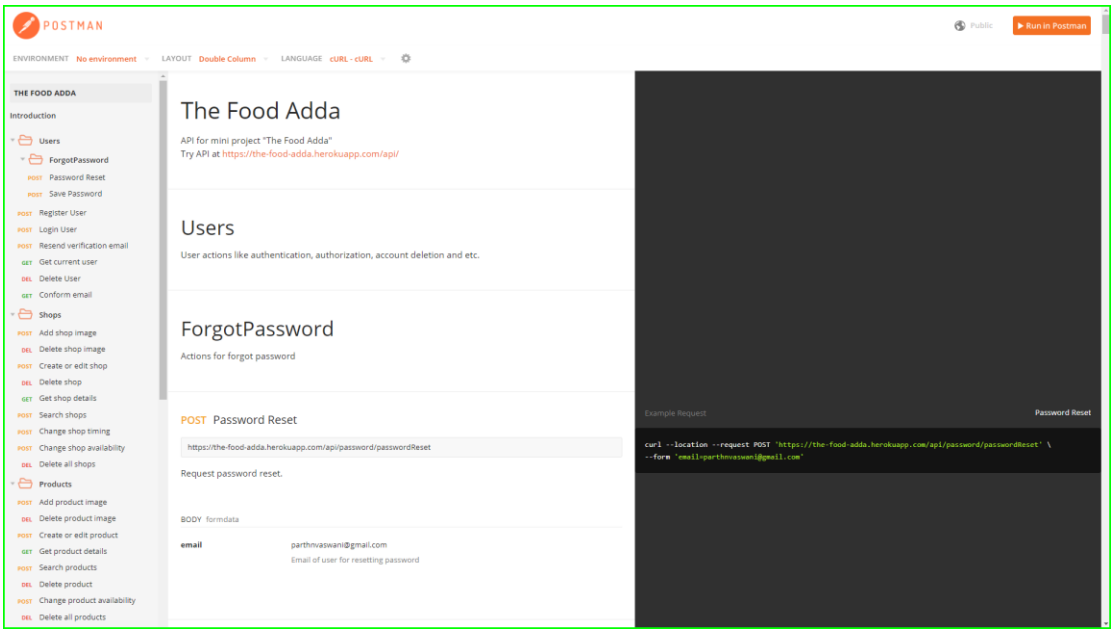
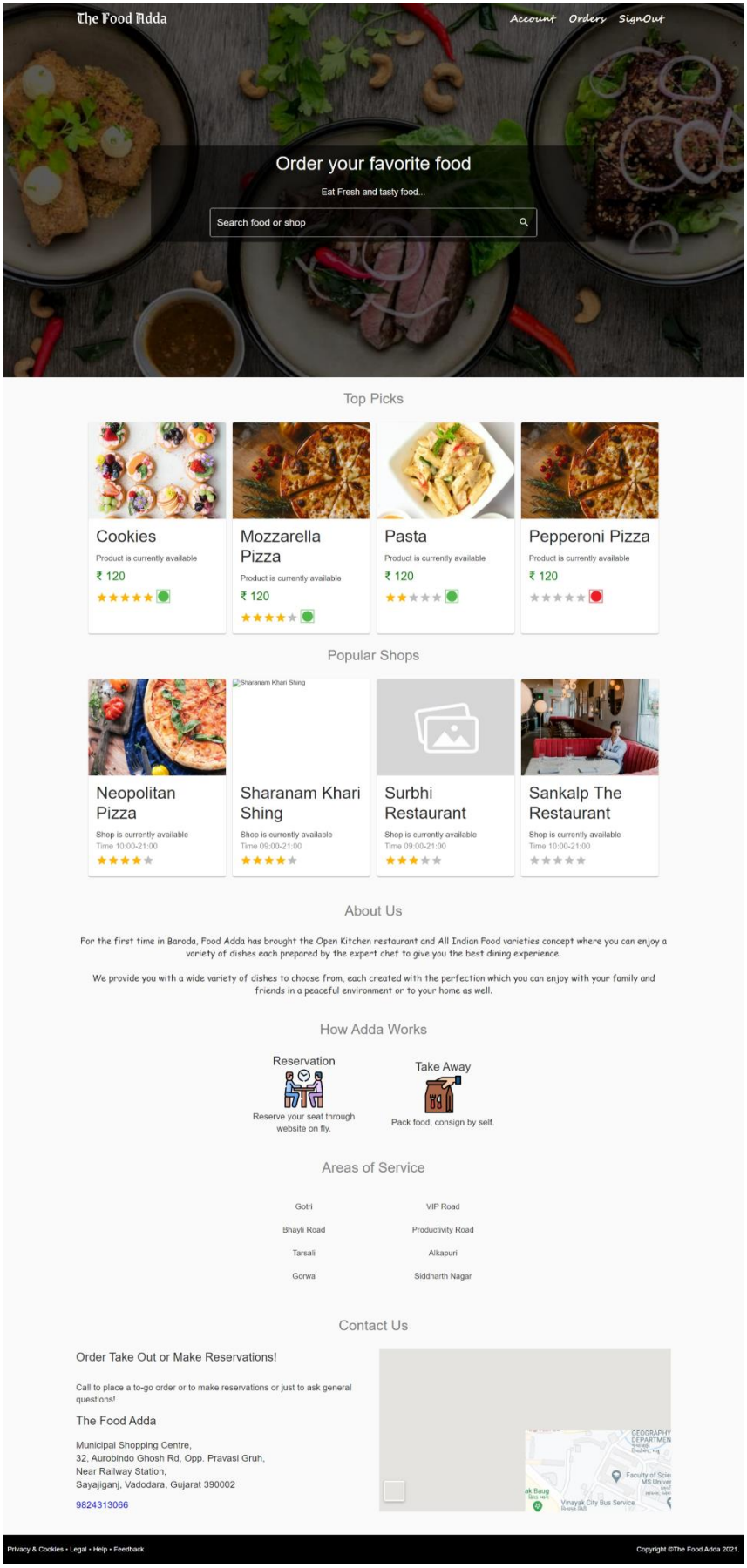


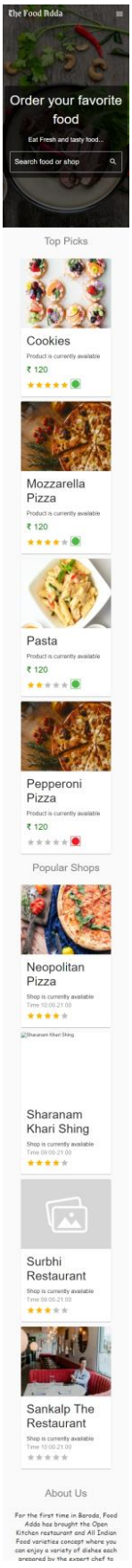
Fig: APIs of "The Food Adda"

Every API is documented and tested in the postman; whose link is given in testing section.

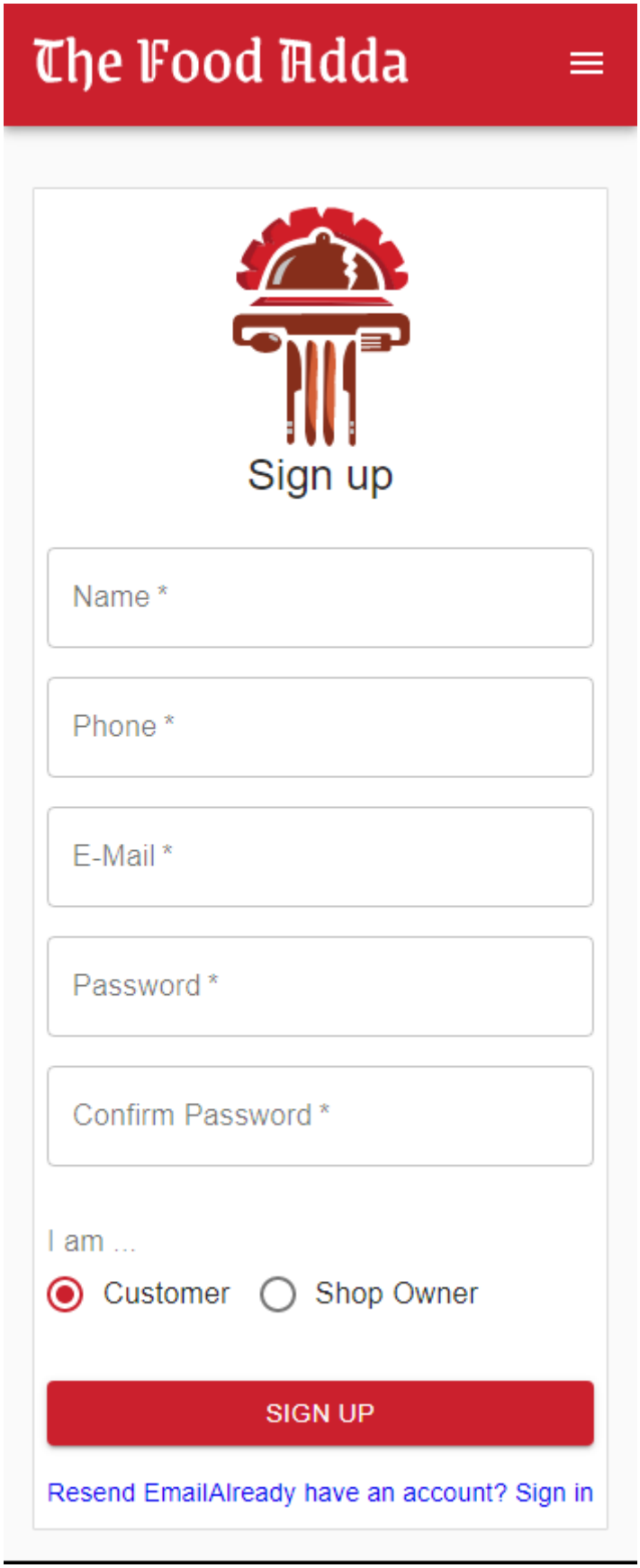


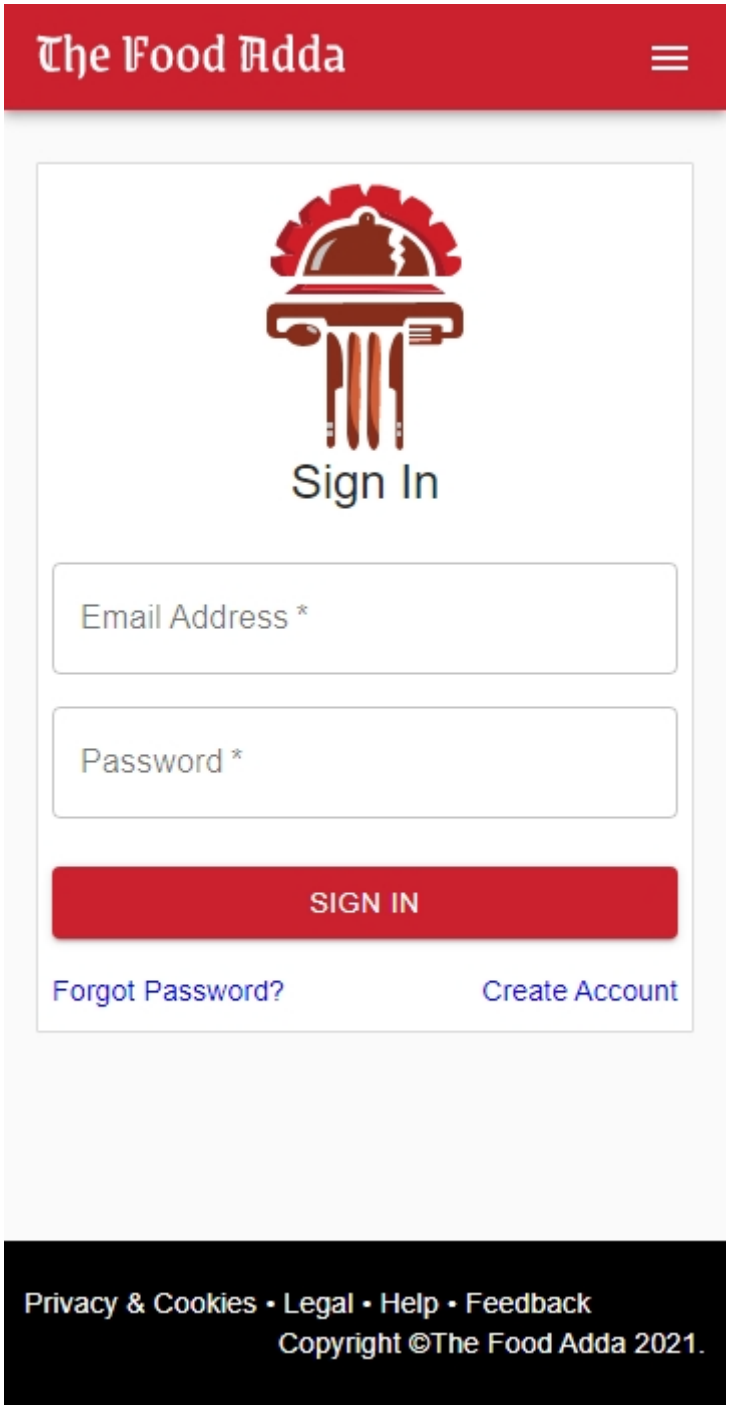
This is landing page of THE FOOD ADDA.

Every page of the application is either adaptive or responsive, which means page can be seen in mobile as well.

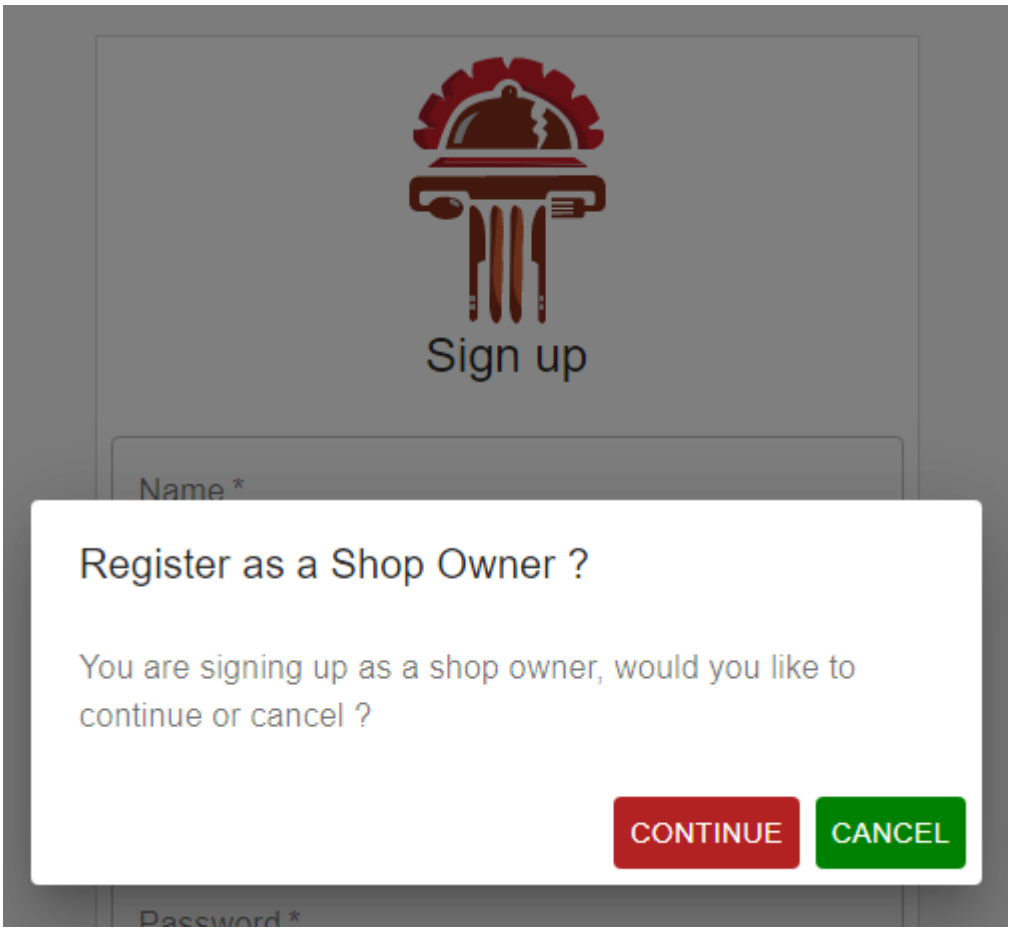


These are sign up and sign in pages, respectively.

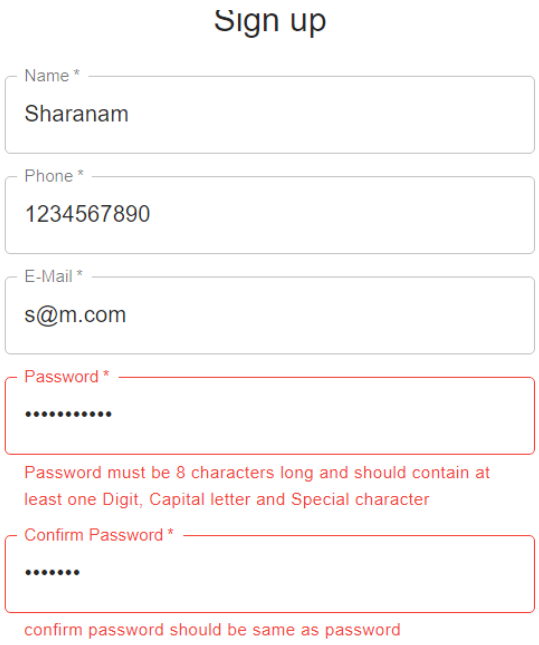



The Food Adda
Sign In
Email Address *
Password *
SIGN IN
Forgot Password?
Create Account
Privacy & Cookies • Legal • Help • Feedback
Copyright ©The Food Adda 2021.

In sign up, if user claims that he is shop owner, we ask him to confirm he is not making mistake.



This is how we displays error….



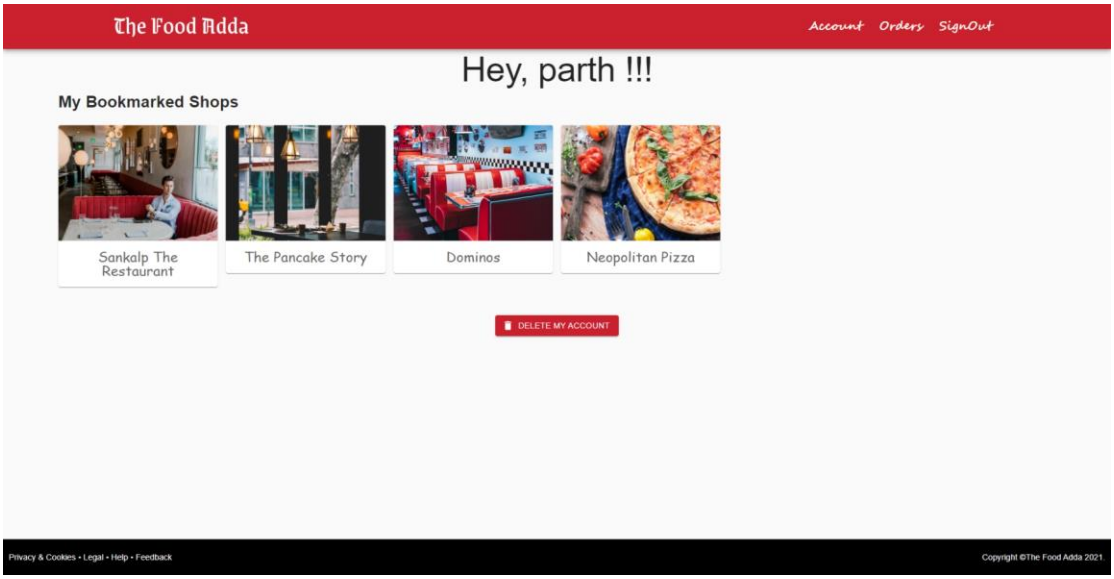


Fig. Account page of Customer, where he is currently watching bookmarked shops.

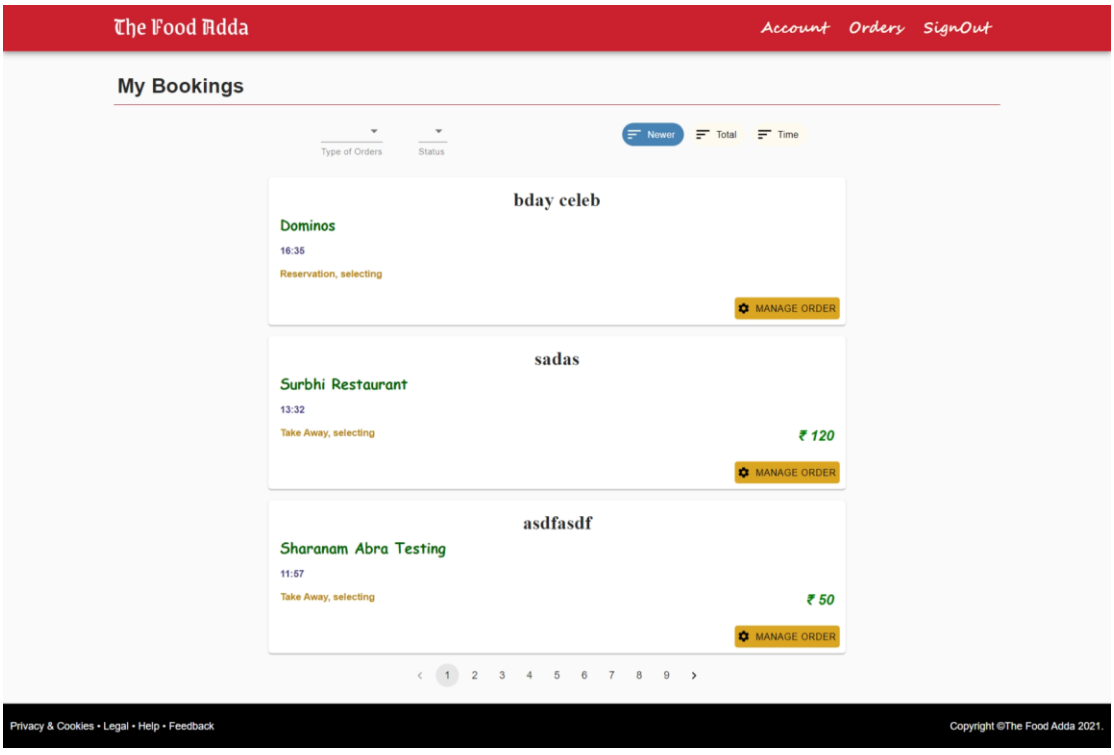


Fig. Orders page of Customer, where he can see his bookings, reservation, and status of each order.

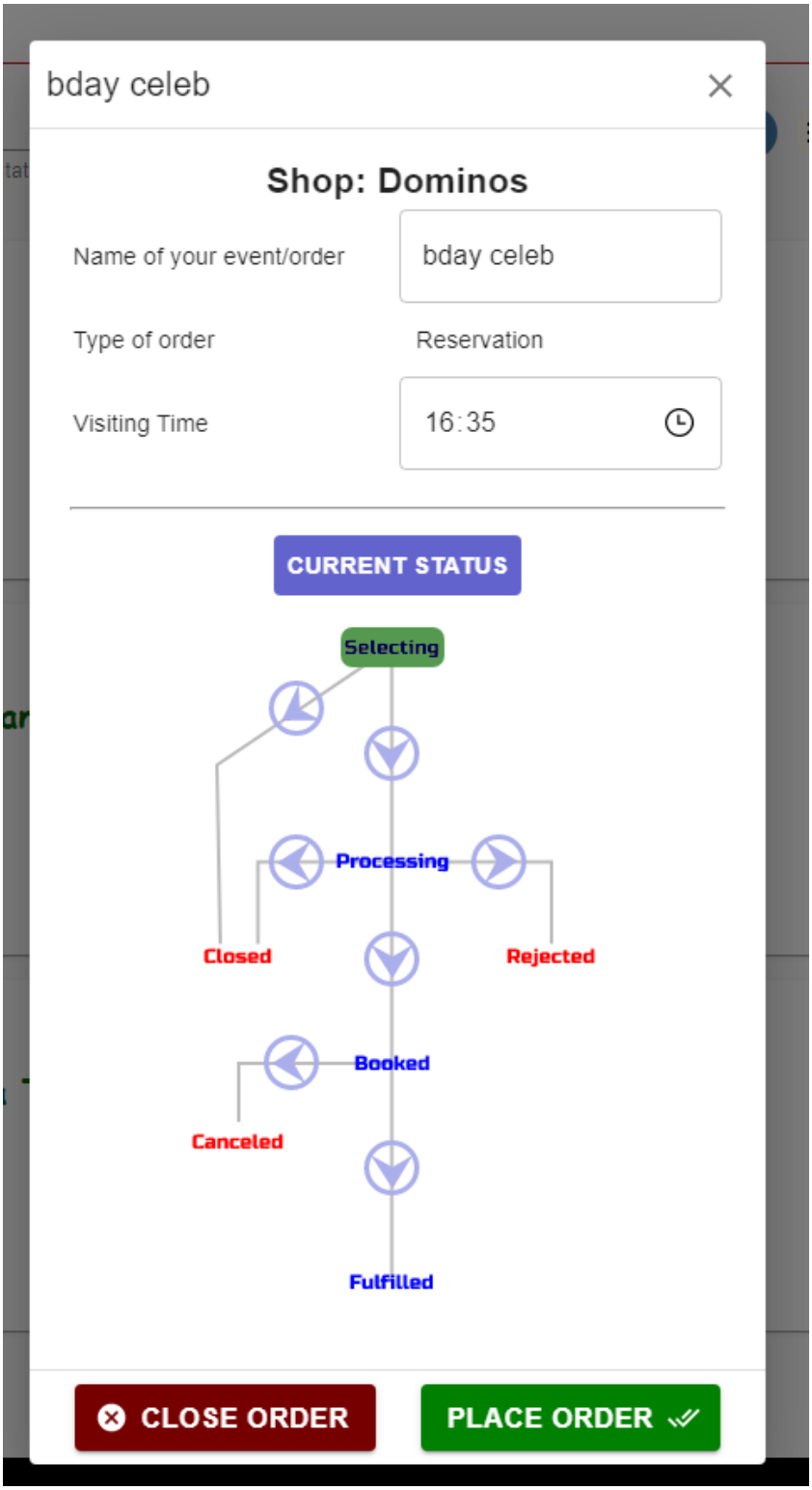


Fig. manage order looks like this, where customer can watch graph of status of order, can change name and time of order, finalize, or cancel the order.

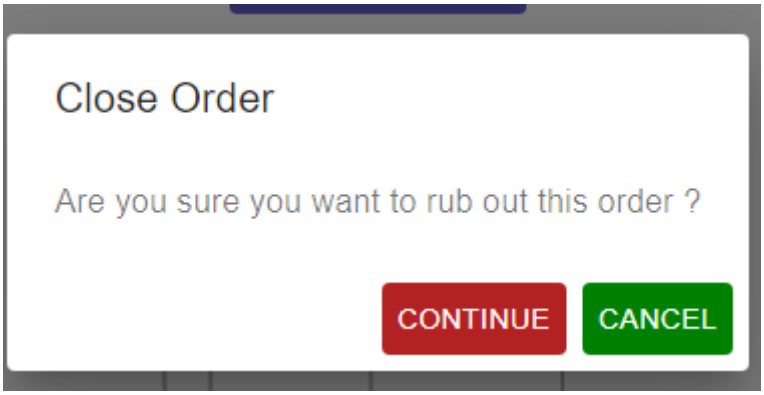


Fig.: confirmation before cancelling the order.

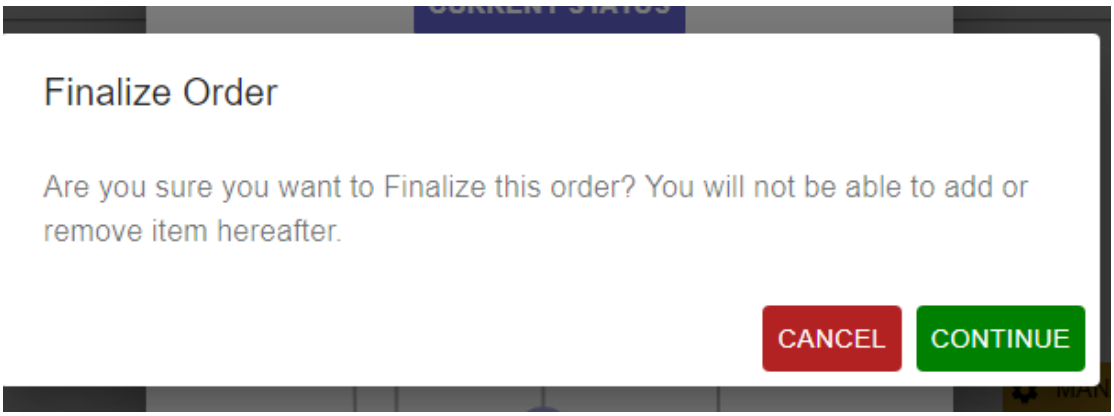


Fig. Confirmation before finalizing order.

If you missed to notice, look at the sequence of "cancel" and "continue". It is required to consider how it will impact to user. If it is impacting negatively, we are putting them inverse, so user cannot click on it by mistake.

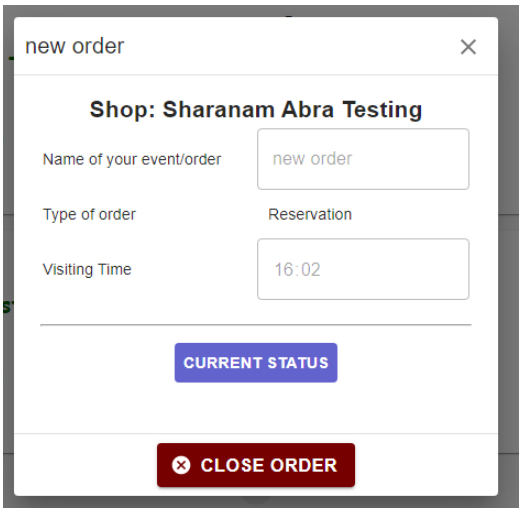


Fig. Showing Order (same dialog box)

Which is finalized by customer. Customer can close this order before shop owner confirms or rejects it. But here customer cannot modify anything afterwards.

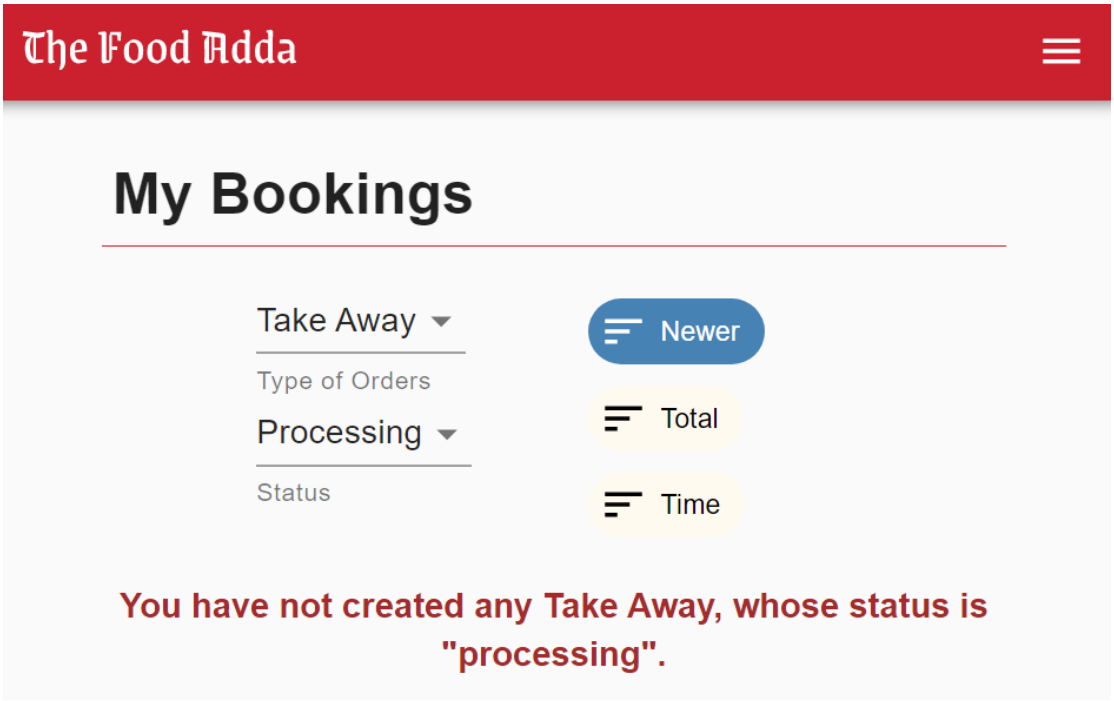


Fig.: displaying error to customer, that he has chosen criteria for which no order matches.

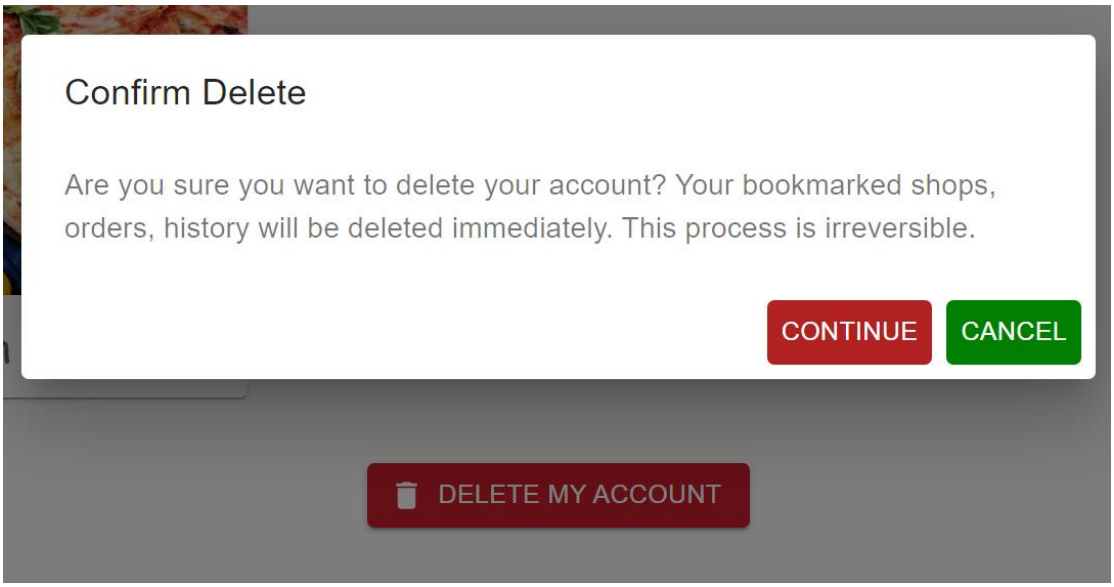


Fig.: Customer wants to delete his account and system asking him to confirm his choice.

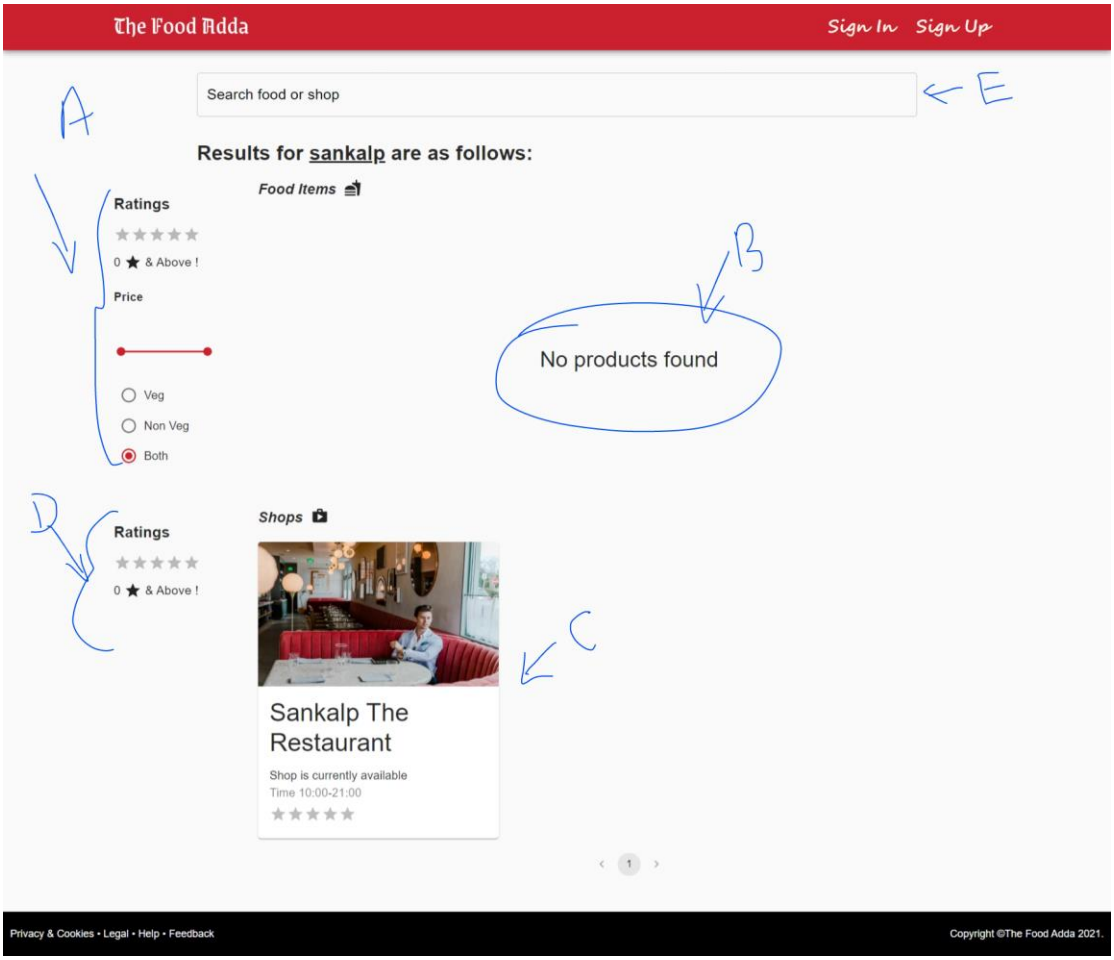


Fig: User is searching for “Sankalp”, we are using regex to fetch results.

In snapshot:

A. Sorting on food products.
B. Area where cards of products will be displayed (i.e., no match found)
C. Are where chards of shops are being displayed.
D. Sorting on shops.
E. Search box for another query.

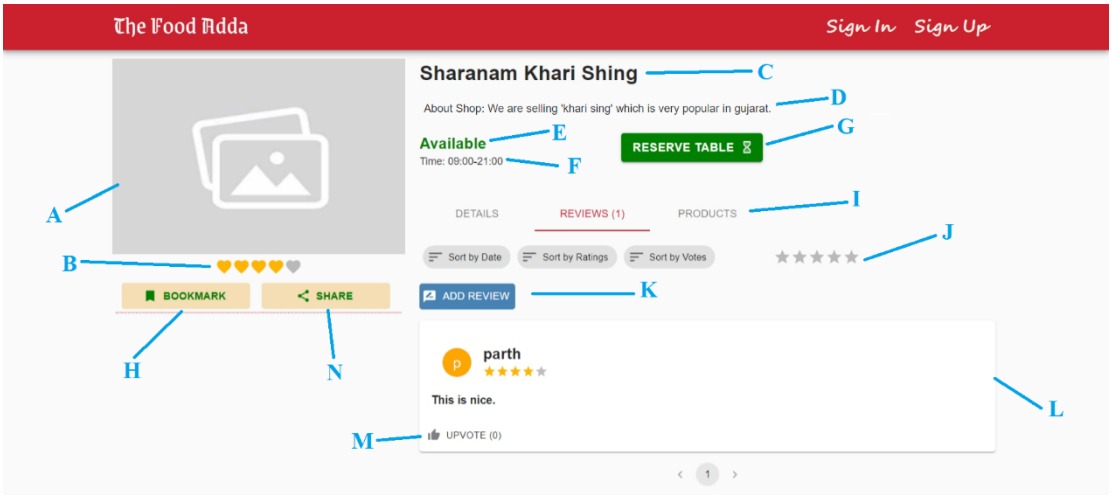


Fig: web page of any given shop.

In snapshot:

A. Image of shop uploaded by owner himself.
B. Displaying rating of shop. It is being resulted by algorithm which accepts all the ratings given from customers.
C. Name of the shop
D. Description of shop written by owner.
E. Whether shop is available or not.
F. Timing of shop
G. Button to reserve table, from where user will be prompted for selecting time.
H. Bookmarking shop, (sign in required by customer)
I. In page navigation, 'reviews' tab currently selected.
J. Sort the review by date, ratings, votes, or by all of them; 'minimum stars' is set 0 by default, if user only wants to see reviews of 4 or 5 stars, he can change it by simply clicking on it. (it is self-descriptive, also we have given tooltip on it)
K. Button to get dialog box to put comment on shop.
L. Displaying card of comments
M. Button to upvote the review of someone else. It indicates that you liked, and comment is useful to you.
N. Share this product on social media. (URL)

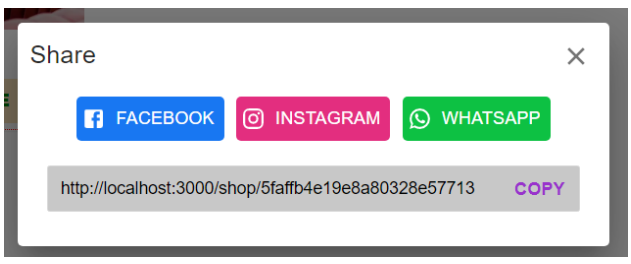


Fig: share link on social media post or copy link

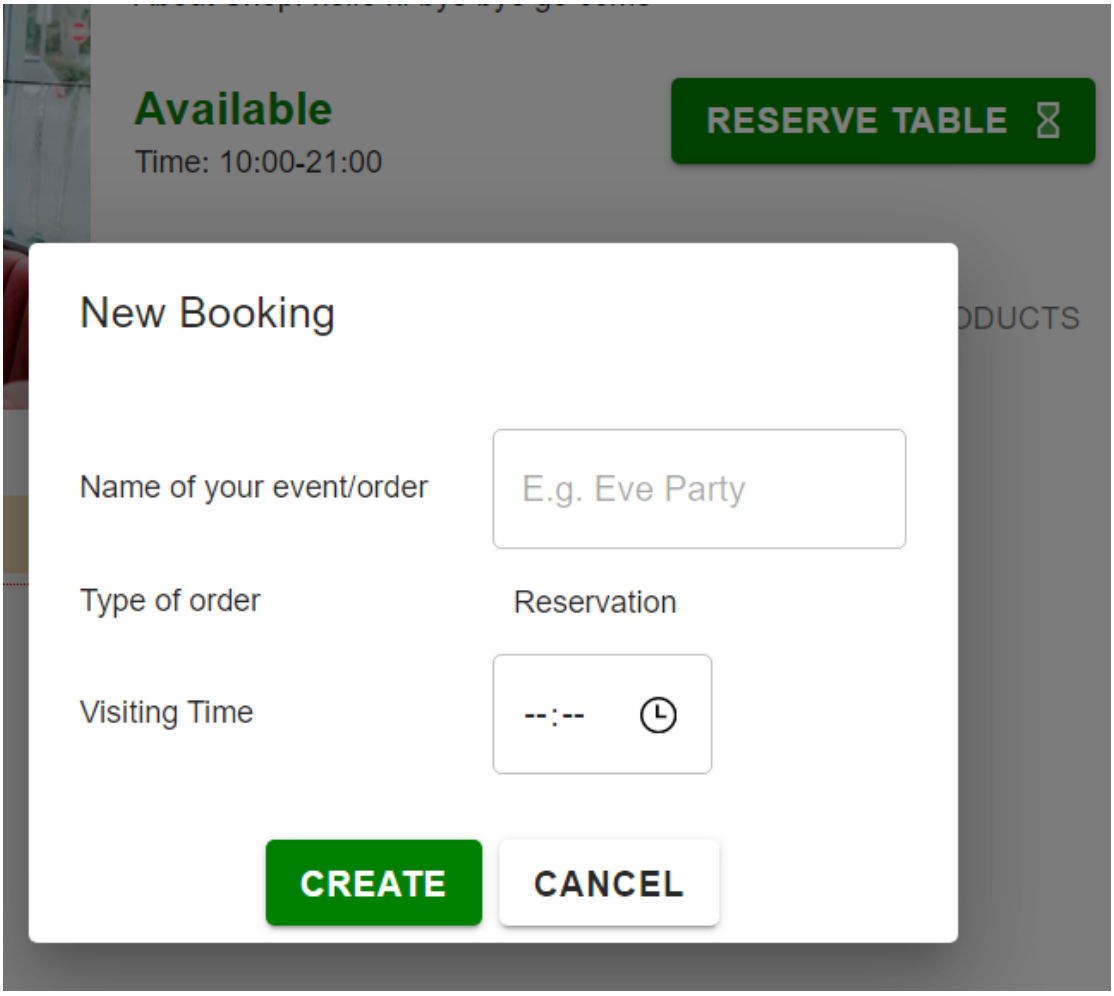
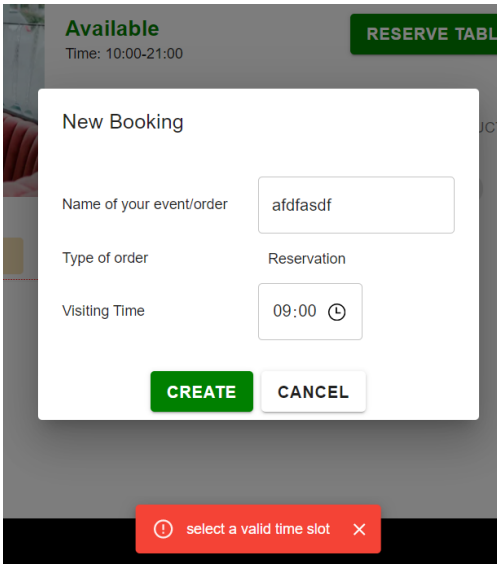


Fig: Reservation of the table. Here order name should be appropriate text of occasion.

Customer will also have to set time according to working time of shop.

Otherwise, he will see error… as follows.



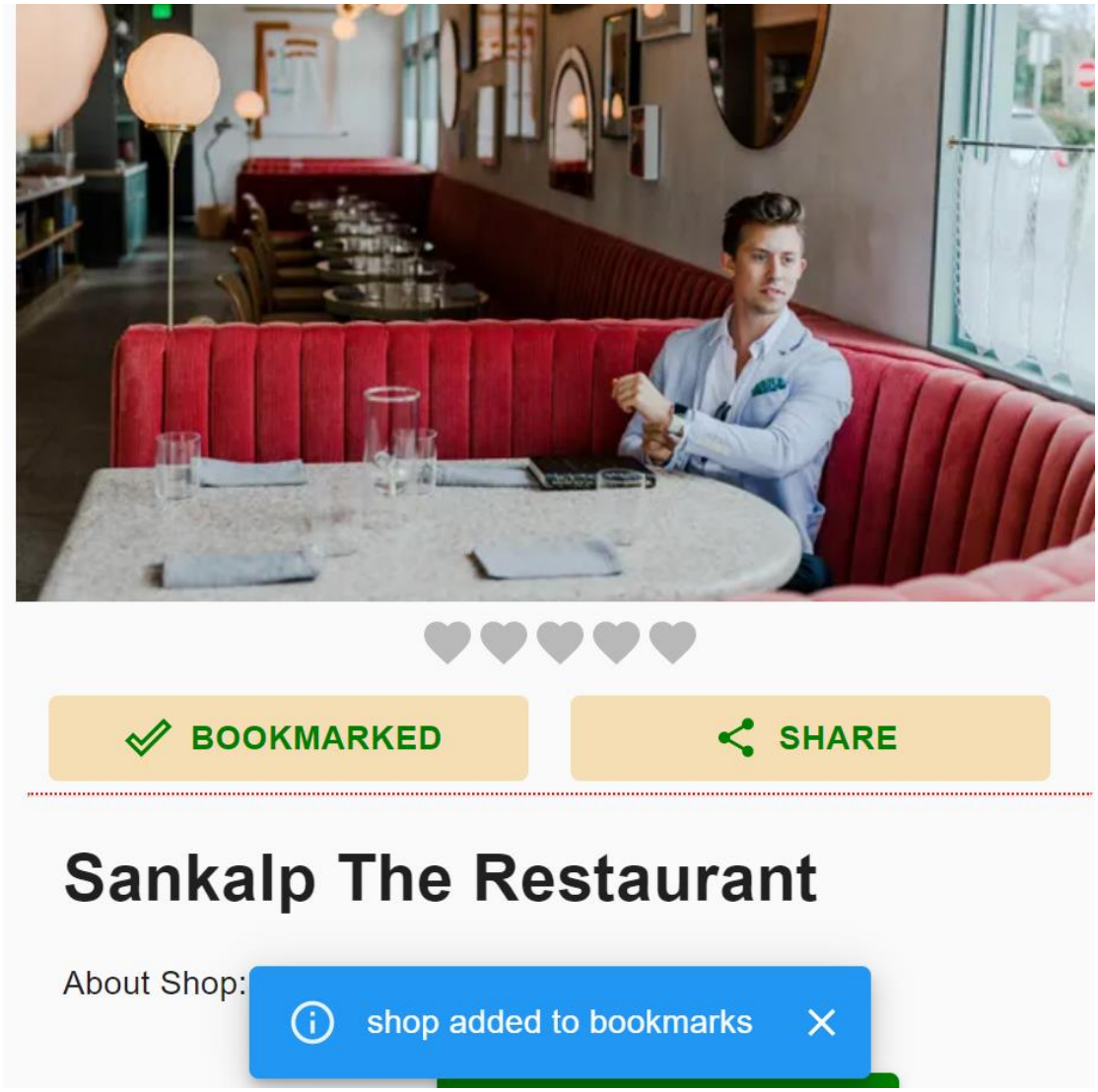


Fig.: Bookmarking the shop

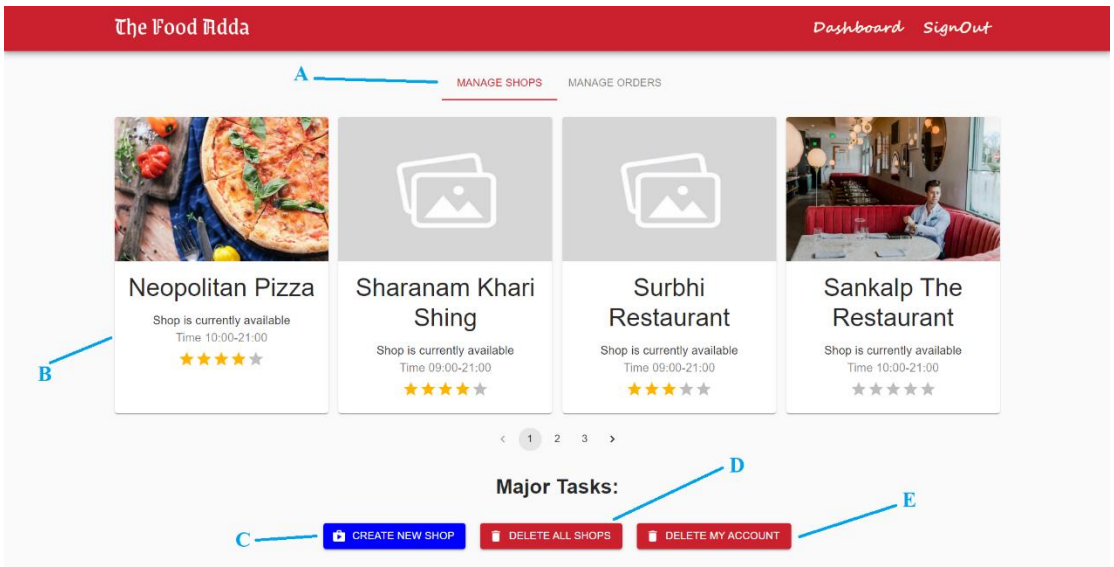


Fig. dashboard of shop owner, where he can manage his shop and orders.

Here,

A. Tabs from where he can navigate between manage shop & products and orders from customers. Shop is currently selected.
B. Shop cards to be managed by clicking on it.
C. Introducing new shop to database.
D. Deleting all the shops from database.
E. Deleting owner's account from system.

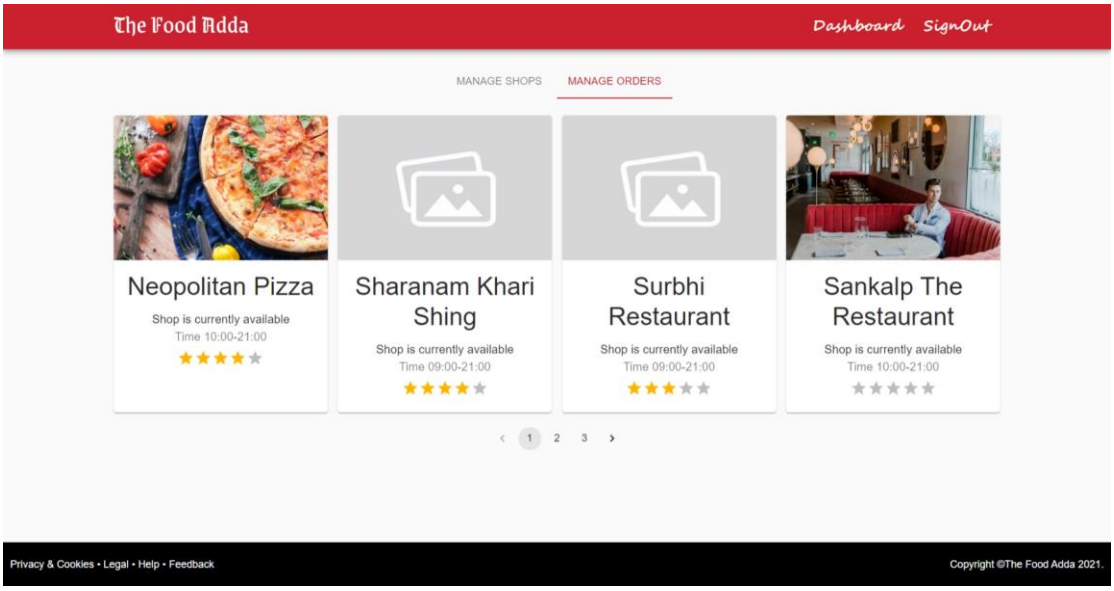


Fig.: managing orders tab selected, where owner can manage orders of any given shop.

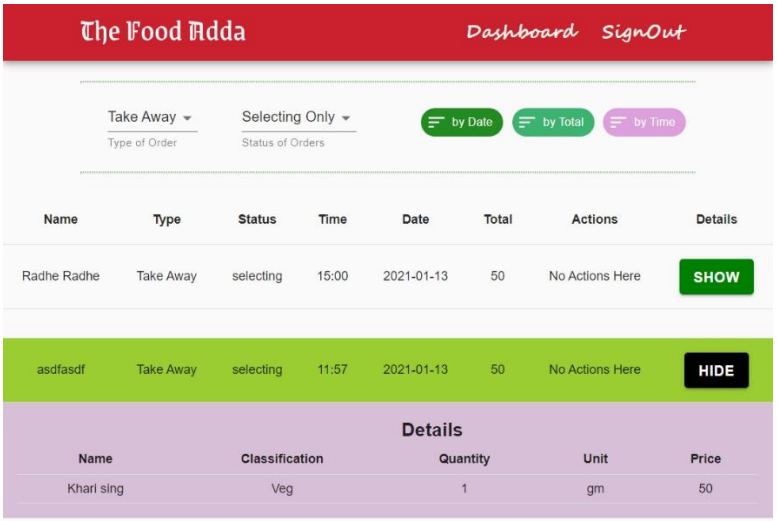


Fig: showing orders made by customer, but not finalized yet.

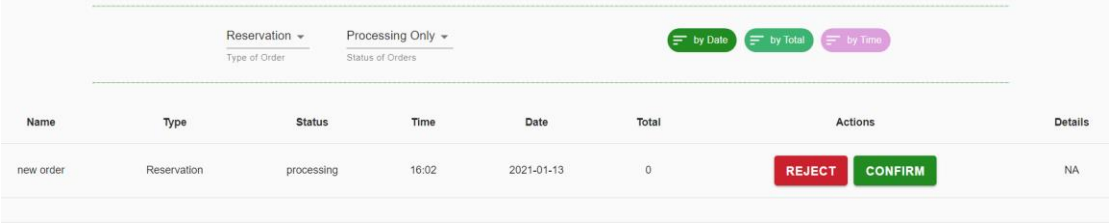


Fig: owner watching and managing orders made by customer.

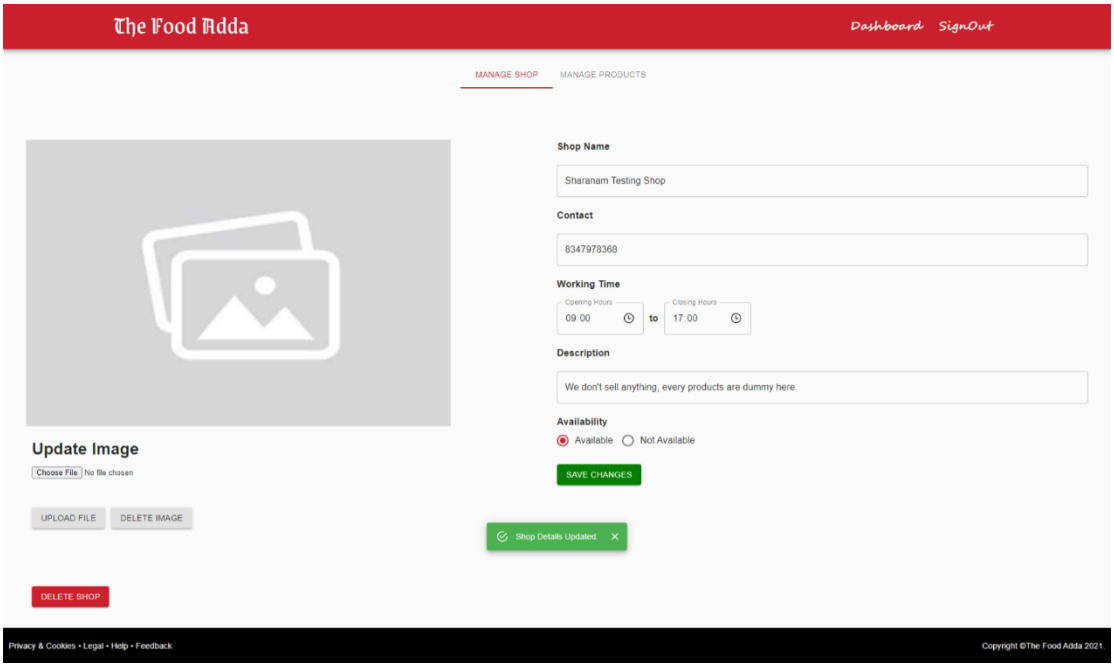


Fig: updating details and image of shop also removing shop from system.

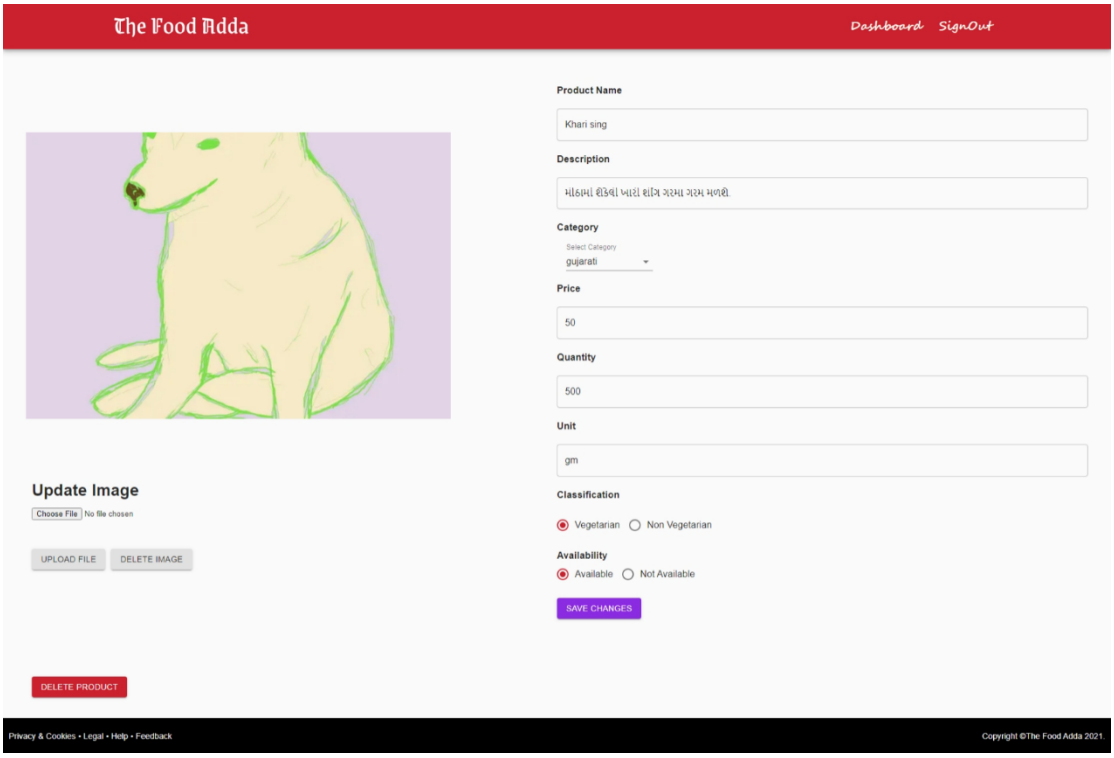


Fig: Managing product details to be shown to customer.

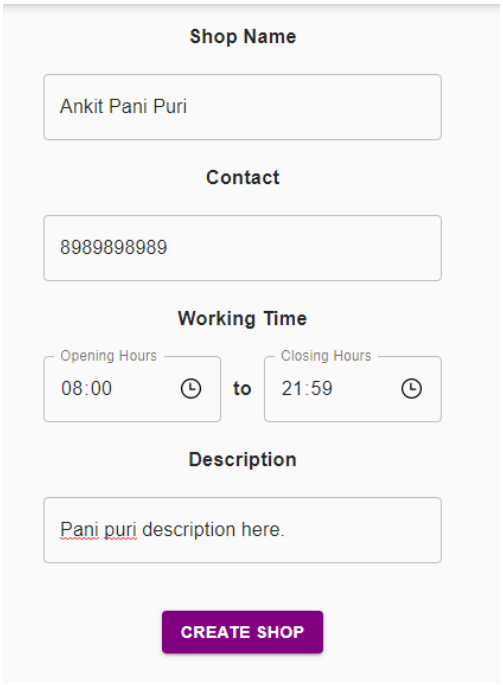


Fig: registering new shop in system.

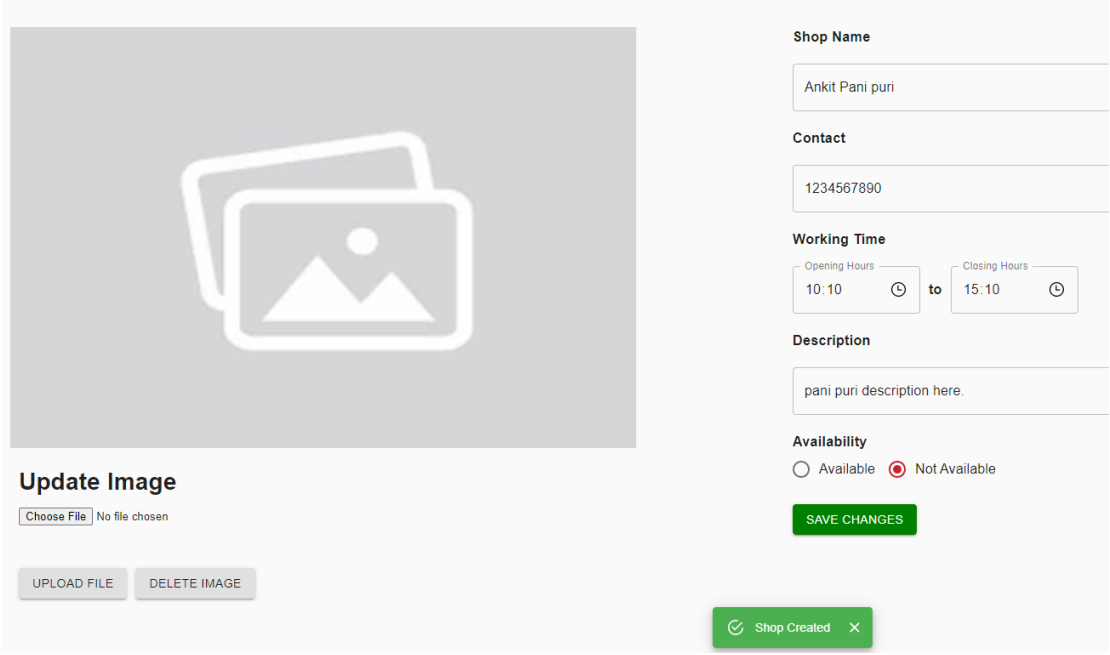


Fig: updating details like availability, image of shop as soon as shop is created.

Other details can be modified as well.

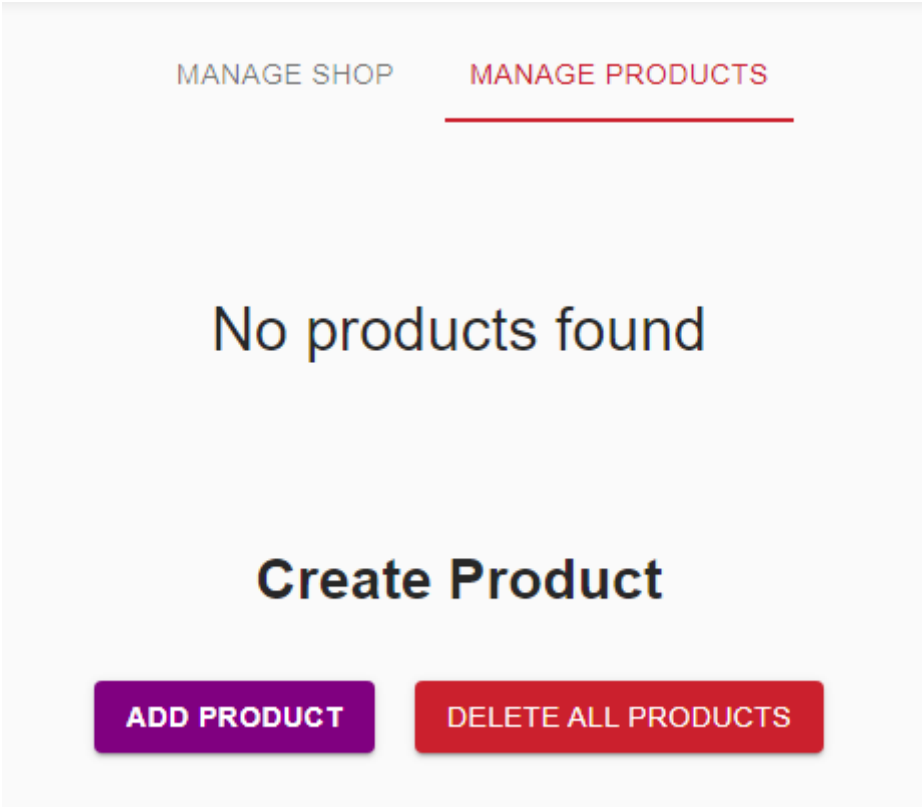


Fig: adding product into shop.(also watch next snapshot)

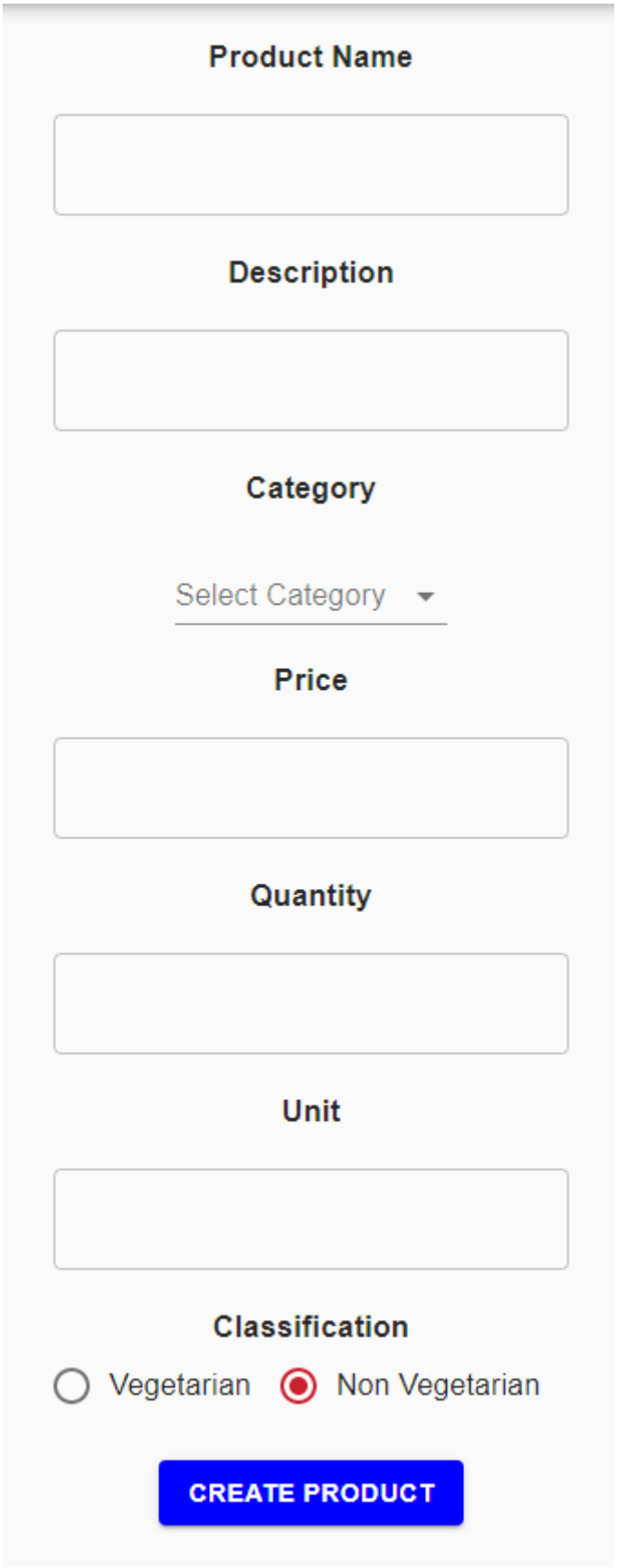


Fig: registering new product

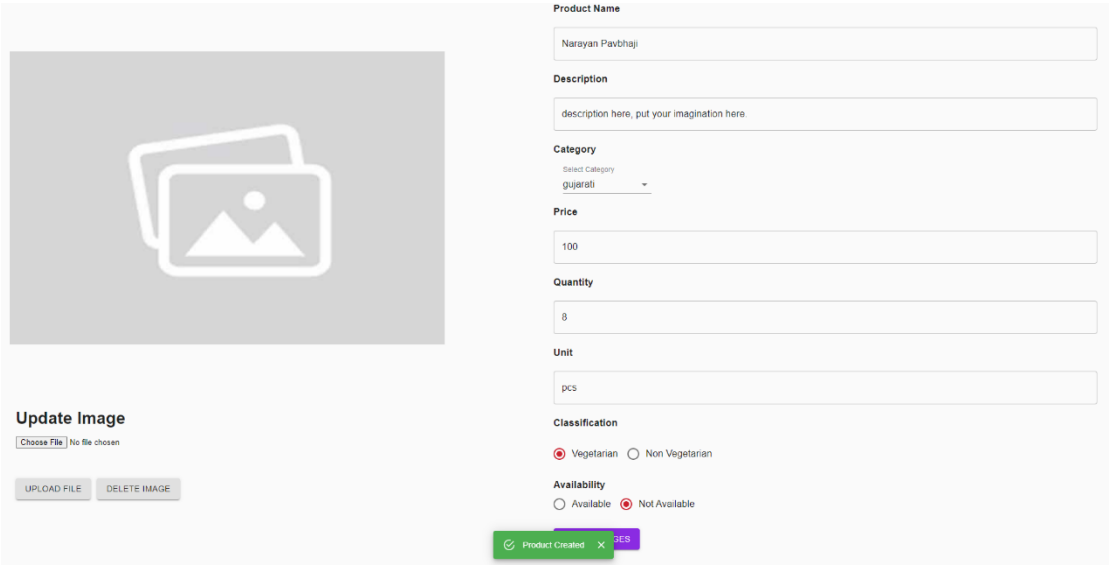


Fig: product is ready to be updated.

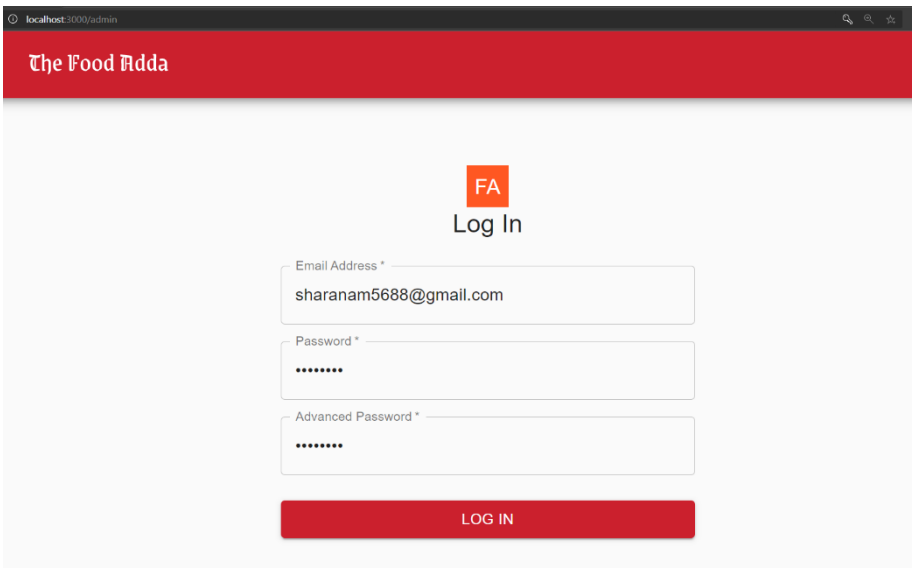


Fig: login page of admin.

Here, path of admin must be written by self, hyperlink is given nowhere.

Admin needs dual password to be logged in. Count it as security feature.

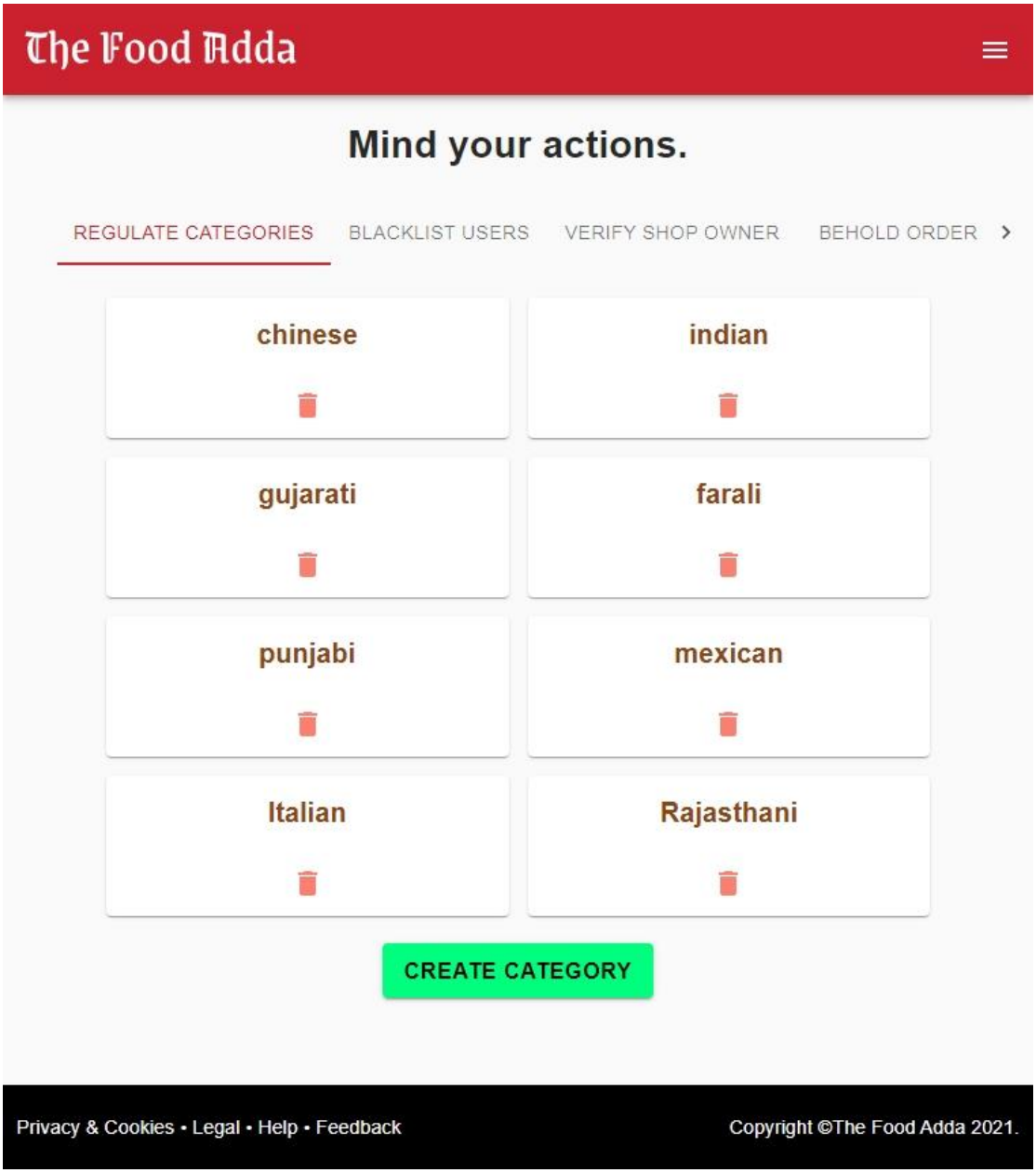


Fig.: landing page of administrator.

Admin can navigate through following tabs:

1. Categories,
2. Users,
3. Shop owners,
4. Orders.

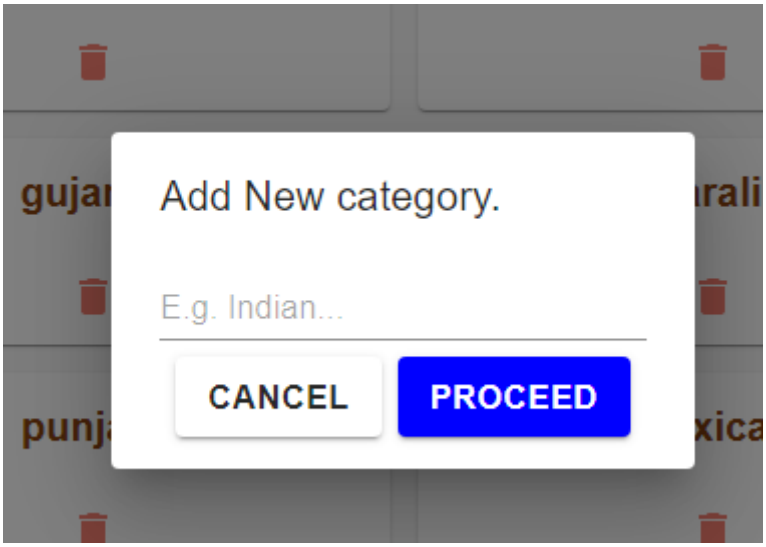


Fig: admin making new category

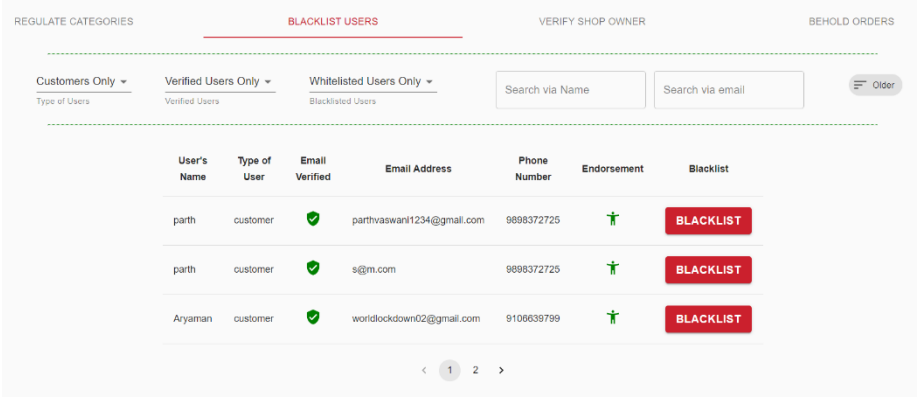


Fig: admin blacklisting or whitelisting users.

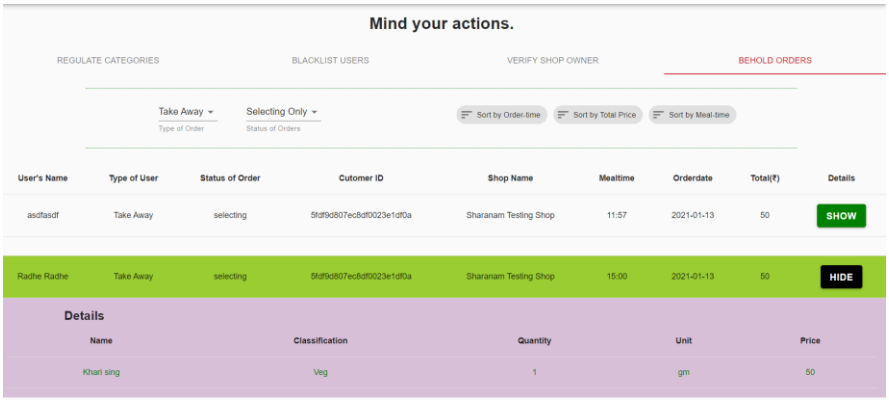


Fig: admin inspecting all the orders.

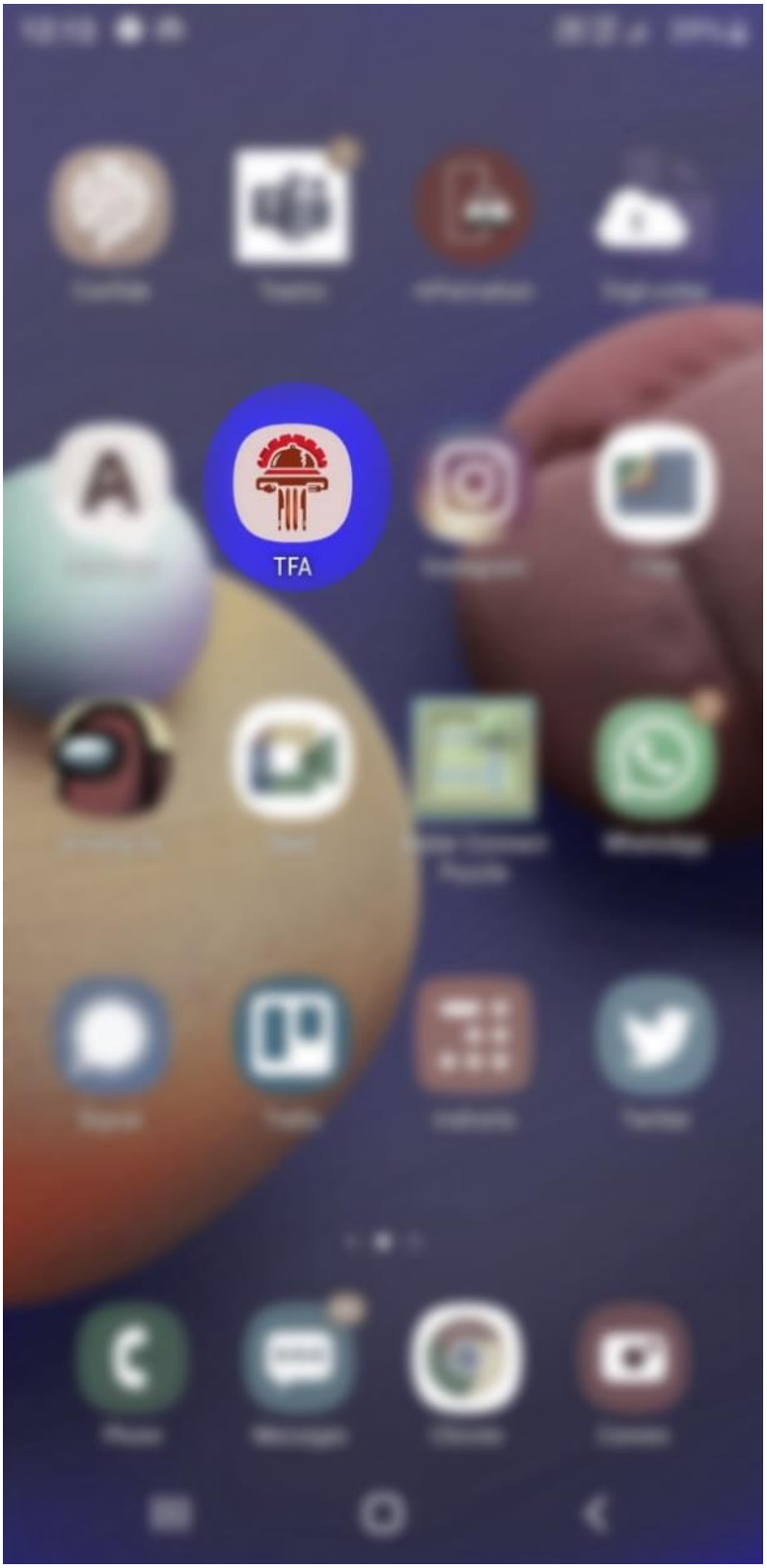


Fig: it is progressive web app, which can be installed in devices like native apps.

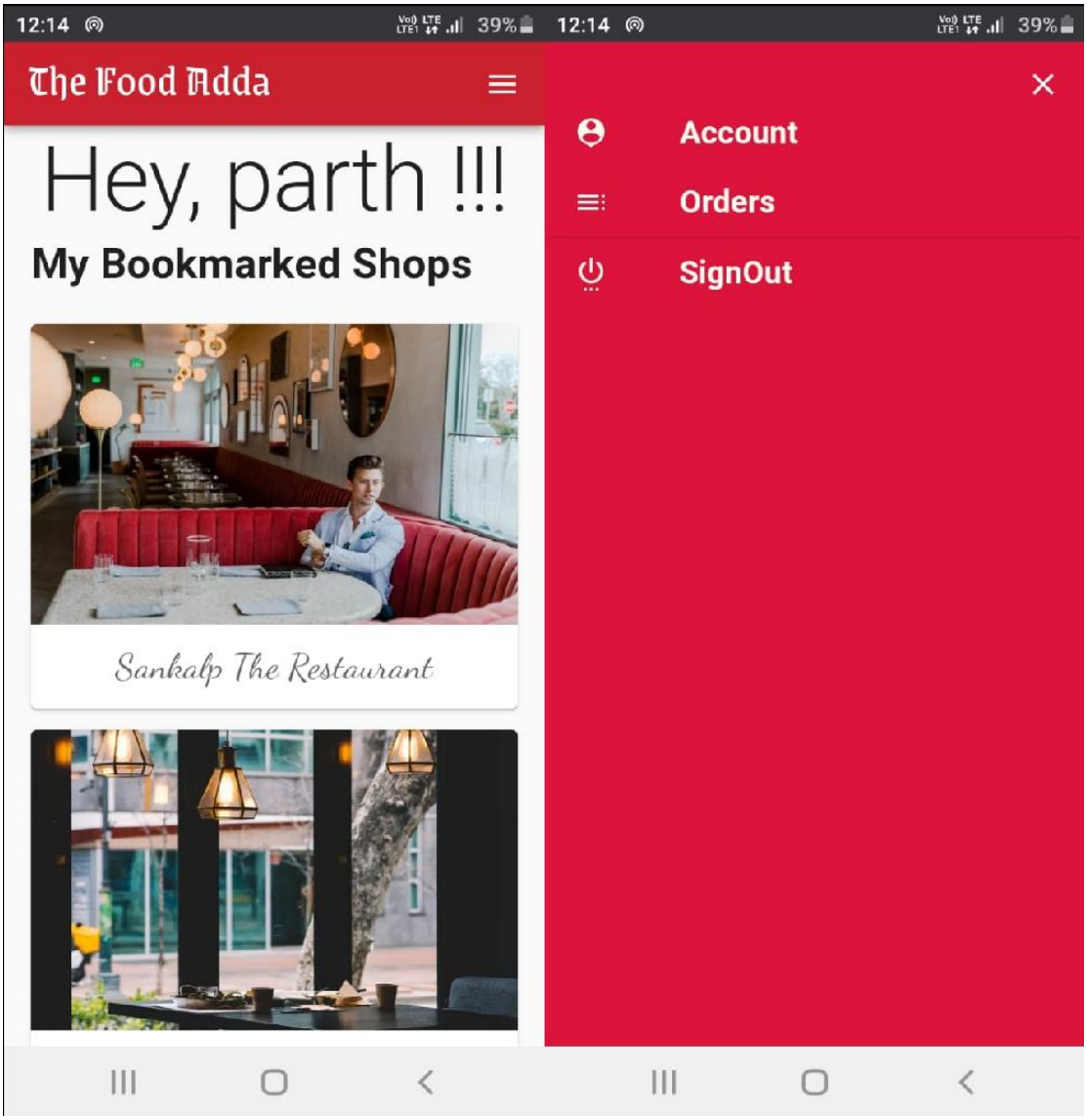


Fig.: running version of application in android device as native application.

Testing

In agile, requirements evolve through collaboration between the customer and self-organizing teams (developers). And agile aligns development with customer needs.

There are few advantages of agile testing:

- It saves time because developers do not need to document everything as 'less documentation' is feature of agile and regular feedback from the end user can help to determine the issues well in advance.
- Developers and end users are testing everything therefore agile do not consists of testing phase in its lifecycle. As testing team is not required, cost reduces.

As mentioned earlier, we were using postman for API testing and API documenting and can be seen and tested on http://the-food-adda.herokuapp.com/. Therefore, we are not adding its copy in this documentation to make it light weight.

Instead of every kind of test that is not being documented in agile methodology, we are putting user scenarios here, which can also be considered as validation, verification, or acceptance test.

User story: "As a user I want sign up form so that I can create new account in system."

Sign Up			
Note	Password must be 8 characters long and should contain at least one Digit, Capital letter and Special character to be said as valid		
	confirm password should be same as password		
	Scene1	**Scene2**	**Scene3**
Name, Phone, Email, Password, Confirm Password	Valid	Valid	Any of the arguments is invalid
Type of user	Customer	Shop Owner	Customer/ShopOwner
Action	by clicking on "sign up" button, send request to server.	by clicking on "sign up" button, confirm before sending request to server.	by clicking on "sign up" button, send request to server.
Output	Display information of "For verifying email ID open the link sent to entered email"	same as scene1	Display error message for specific inputbox

User story: "As a user I want sign in form, so that I can get into system."

Log In (shopowner & customer)						
Note	User must have been signed up and have email verified to pass it.					
	Type of user is checked by server itself, and will send details in response to client.					
	Scene1	**Scene2**	**Scene3**	**Scene4**	**Scene5**	**Scene6**
Username	Valid	Valid	Invalid	Valid	Valid	Valid
Password	Valid	Invalid	Valid	Valid	Valid	Valid
Signed Up	Yes	Yes	Yes	No	Yes	Yes
Email Verified	Yes	Yes	Yes	No	No	Yes
Type of User	Customer	Customer	Customer	Customer	Customer	Shop Owner
Action	by clicking on "sign in" button, send request to server.					
Output	Store Token in localhost and Redirect to Account page	Display error received from server	Display error received from server	Display alert received from server	Display alert received from server	Store Token in localhost and Redirect to shop owner's dashboard

Delete Account	
Note	User must be signed up
Action	by clicking on "delete my account" button, send request to server only after confirming through dialog box.
Output	Clear user details and personalized information through back end

---

User stories: “As a site visitor, I want Homepage so that I can get overview of your services.”, “As a Site Visitor, I want Navigation bar so that I can view different pages.”

Redirect to homepage	
Note	From any page of the website, user can click on "The Food Adda" logo
Action	by clicking on logo,
Output	User will be redirected to homepage/landing page

User story: “As a Customer, I want search box in landing page, so that I can search my product or any shop conveniently.”

Search Box Test Case			
Note	Authentication is not required.		
	Scene1	**Scene2**	**Scene3**
input text	"t"	"Sankalp"	"kalp" to get sankalp as output
Action	send query to server on enter key press	send query to server on enter key press	send query to server on enter key press
Output	Redirect to search page, with products and shops, which contains "t" in their name.	Redirect to search page, with products and shops, which contains "Sankalp" in their name.	Redirect to search page, with products and shops, which contains "kalp" in their name. Sankalp is in the list.

User Stories: “As a Shop Owner, I want one dashboard for all the shops so that I can manage.”, “As a shop owner I want a page that can help me manage all orders and

bookings made to my shop.”

Shop Owner

Manage Shop Details	
Prerequisite	Owner has created shop
	Scene1
Shop Name, Contact, Working Time, Description, Availability	Update Value of any of the parameters
Action	click on save changes button
Output	Display success message

Manage Shop Image		
Prerequisite	Owner has created shop	
	Scene1	**Scene2**
Image	Upload (change) image	Delete image
Action	Choose image and hit "Upload Image" button	Press "Delete Image" button and confirm
Output	Display updated image onto page	Display "Null" image onto page

Manage Products Details	
Prerequisite	Owner has created shop
	Scene1
Product Name, Description, Category, Price, Quantity, Unit, Classification, Availability	Update Value of any of the parameters
Action	click on save changes button
Output	Display success message

Manage Products Image		
Prerequisite	Owner has created product	
	Scene1	**Scene2**
Image	Upload (change) image	Delete image
Action	Choose image and hit "Upload Image" button	Press "Delete Image" button and confirm
Output	Display updated image onto page	Display "Null" image onto page

Delete Product	
Prerequisite	Shop owner must be on product page
	Scene1
Action	Press "Delete Product" button
Output	Redirect to shopowner's dashboard, after confirming.

Delete Shop	
Prerequisite	Shop owner must be on shop page
	Scene1
Action	Press "Delete Shop" button
Output	Redirect to shopowner's dashboard, after confirming.

Add Products		
Note	Shop owner has atleast one shop, in which he is adding product.	
	After successfully creation of product, owner can upload image for respective	
	Scene1	**Scene2**
Product Name, Description, Category, Price, Quantity, Unit, Classification	Enter Valid Details in each field	Enter invalid Details in each field
Action	Proceed to "Create Product"	Proceed to "Create Product"
Output	Message of Shop is created, redirected to manage product page.	Displayed error message

Delete All Products	
Prerequisite	Shop owner must have atleast 1 product in respective shop
	Scene1
Action	Press "Delete All Products" button and confirm
Output	Redirect to shopowner's dashboard

Create New Shop		
Note	After successfully creation of product, owner can upload image for respective	
	Scene1	**Scene2**
Shop Name, Contact, Working Time,	Valid Details in each field	Invalid Details in each field
Action	Proceed to "Create Product"	Proceed to "Create Product"
Output	Message of Shop is created, redirected to manage shop page.	Displayed error message

Delete All Shops	
Prerequisite	Shop owner must have atleast 1 shop
	Scene1
Action	Press "Delete All Shops" button and confirm
Output	Redirect to shopowner's dashboard

Manage Orders			
Prerequisite	Shop owner has shop, and orders received from customers		
	Scene1	**Scene2**	**Scene3**
To-do	Confirm Order	Reject Order	Fulfill Order
Action	on "Confirm" Button Click	on "Reject" Button Click	on "Fulfill" Button Click
Output	Confirm through dialog box and change status of order to confirmed	Confirm through dialog box and change status of order to rejected	Confirm through dialog box and change status of order to fulfilled

Customer

Add to Cart		
Prerequisites:	quantity is seleted and User is signed in as customer	
	Scene1	**Scene2**
Note	(User has no booking for product of its shop)	(User has already order in selecting state for product of its shop)
New Booking	Yes	No
Output	Displayed Creation of New Book dialog box	Appended product in order (cart) with notification

User stories: “As a Customer, I want to reserve a table for a specific time slot to use their services.”, “As a Customer, I want to order a food online to take it's delivery.”, “As a customer, I want a decent UI to book and place my order for any specific shop or product.”

New Booking		
Prerequisites:	Customer has followed 'Add to Cart' Procedure	
	Scene1	**Scene2**
Order type is	(Parcel or Take Away)	(Table)
Name of order	"New Year Party"	"get together"
Visiting Time	In between of shop's working time	In between of shop's working time
User Action	"Create" booking	"Create" booking
Output	Redirected to Orders Page	Dialog box closed with notification

User story: “As a Customer, I want a page where I can manage my pending booking as well as orders.”

Manage Order						
Prerequisites:	Customer is signed in and viewing orders page					
	Scene1	**Scene2**	**Scene3**	**Scene4**	**Scene5**	**Scene6**
Note	Order type is Take Away	Order type is Take Away				
To-do	Change the quantity of selected product	Remove item from cart	Close/cancel Order	Place Order	Rename Order	Change Visiting Time
User Action	Select input box of appropriate item, and change	"Create" booking	Press "Cancel Order" and confirm	Press "Finalize Order" and confirm	Modify Text in input box and hit enter	Modify Visiting Time and hit enter
Output	Updated quantity and change colors during process	Confirmed through message box and removed item, displayed notification	Change status of order to closed or canceled, as per the previous state of order	Change status of order to processing	Updated name of order	Updated visiting time

Manage Bookmark		
Prerequisite	Customer is signed in	
	Scene1	**Scene2**
shop is bookmarked already	Yes	No
User Action	Press "Bookmark" button	Press "Bookmark" button
Output	Notification "Removed from bookmarks"	Notification "Added into bookmarks"

User story: “As a Customer, I can view other people's opinion i.e., reviews and put my reviews for specific product or shop. Moreover, I can also vote someone's opinion if I like it or can withdraw my vote if I didn't.”

Add Review			
Note	User needs to be signed-in		
	User has not reviewed already		
	Scene1	**Scene2**	**Scene3**
user action	Click on the "add review" button	Click on the "add review" button	Proceed to review
user is signed in	No	Yes	Yes (as shop owner)
Output	Redirected to sign in page	Open dialog box for review and ratings	Display error message

Edit Review	
Note	User is signed in, and given review already.
	Scene1
user action	Click on the "edit review" button
Output	Open dialog box for review and ratings

Upvote Review			
Note	User must signed-in		
	Scene1	**Scene2**	**Scene3**
user action	Click on the "upvote" button	Click on the "upvote" button	Click on the "upvote" button
Signed in	Yes	No	Yes
upvoted already	No	Not possible to check (by default No)	Yes
Output	Acknowledge upvote button	Displayed error box "Sign in required"	Play Down Upvote Button

User Stories: “As an Admin, I want dashboard for management so that I can manage all the shop owners at single place.”,” As an Admin, I want a sign in page to access

my account.”

Admin

Log In (Admin)		
	Scene1	**Scene2**
Username, Password, Adv Password	Valid	Any of the arguments is invalid
Action	by clicking on "sign in" button, send request to server.	
Output	Store Token in localhost and Redirect to admin panel	Display error received from server

Regulate Categories		
	Scene1	**Scene2**
Task	Add Category	Delete Category
User Action	Name Category and Proceed	Press remove button of specific card
Output	Category is displayed in list	Category disappeared

Blacklist Users		
	Scene1	**Scene2**
Task	Blacklist user	Whitelist user
User Action	hit "blacklist" button and confirm	hit "whitelist" button and confirm
Output	User disappeared from whitelist users	User is enlisted in whitelist users

Verify Shop Owner	
	Scene1
Task	Verify Shop Owner
User Action	hit "verify" button and confirm
Output	shop owner is now verified message displayed

APIs of THE FOOD ADDA

Users:

(POST) Password Reset

(POST) Save Password

(POST) Register User

(POST) Login User

(POST) Resend verification email

(GET) Get current user

(DEL) Delete User

(GET) Conform email

Shops:

(POST) Add shop image

(DEL) Delete shop image

(POST) Create or edit shop

(DEL) Delete shop

(GET) Get shop details

(POST) Search shops

(POST) Change shop timing

(POST) Change shop availability

(DEL) Delete all shops

Products:

(POST) Add product image

(DEL) Delete product image

(POST) Create or edit product

(GET) Get product details

(POST) Search products

(DEL) Delete product

(POST) Change product availability

(DEL) Delete all products

Customers:

(POST) Bookmark Shop

(DEL) Unbook mark Shop

(GET) Get bookmarked shops

(GET) Check if shop is bookmarked

Admin:

(POST) Login admin

(GET) Get categories

(POST) Add category

(DEL) Delete category

(POST) Get users

(GET) Get unverified shop owners

(POST) Blacklist user

(POST) Verify shop owner

(DEL) Delete shops

Reviews:

(POST) Create or edit review

(POST) Add vote

(DEL) Delete vote

(POST) Search reviews

Bookings:

(POST) Get customer orders

(POST) Get all orders

(POST) Get owner orders

(PUT) Fulfil order

(PUT) Confirm or reject order

(DEL) Remove item

(POST) Add item

(PUT) Change item quantity

(PUT) Cancel order

(PUT) Close order

(PUT) Finalize order

(POST) Create or edit order

Security and Other Features

- XSS
  - xss-clean: Node.js middleware to sanitize user input coming from POST body, GET queries, and url params.
  - Manual Validation: Before storing user input to the database.
  - Helmet: Ensures that no inline or 3rd party are executed on front end by setting some headers.
  - React: Filters the data coming from server before rendering it on front end.
- Regex DoS
  - User input is filtered before inserting it into Regular Expression.
- Denial of Service & Brute force
  - Express Rate Limit: Basic rate-limiting middleware for Express. Use to limit repeated requests to public APIs and/or endpoints such as password reset.
- NoSQL Injection
  - Express Mongoose Sanitize: Middleware which sanitizes user-supplied data to prevent MongoDB Operator Injection.
- CSRF (Cross Site Request Forgery)
  - We are using JWT tokens stored in local-storage which can only be used on the same site-scripts to validate user. And we have already mitigated XSS so, CSRF is not possible.
- Caching
  - The assets are cached on initial load of website using service worker, so that user do not have to wait for website to load every-time they surf it. And it also works offline.
- Installable
  - Our application is installable on any platform, so that they can access it from home-screen without opening any browser.
- Single Page Application (React)
  - It is single page application, so user do not have to wait for the loading of a new page because only content of the page is changed, and data consumption is low.
- Image Compression
  - Every image is compressed and changed to more globally accepted format like JPEG before it is stored.
- Cuss word filter
  - Cuss words are filtered from the reviews, given by the users of site.
- Email Verification
  - On sign up, user will be verified through email, so that only genuine users are able to sign up.

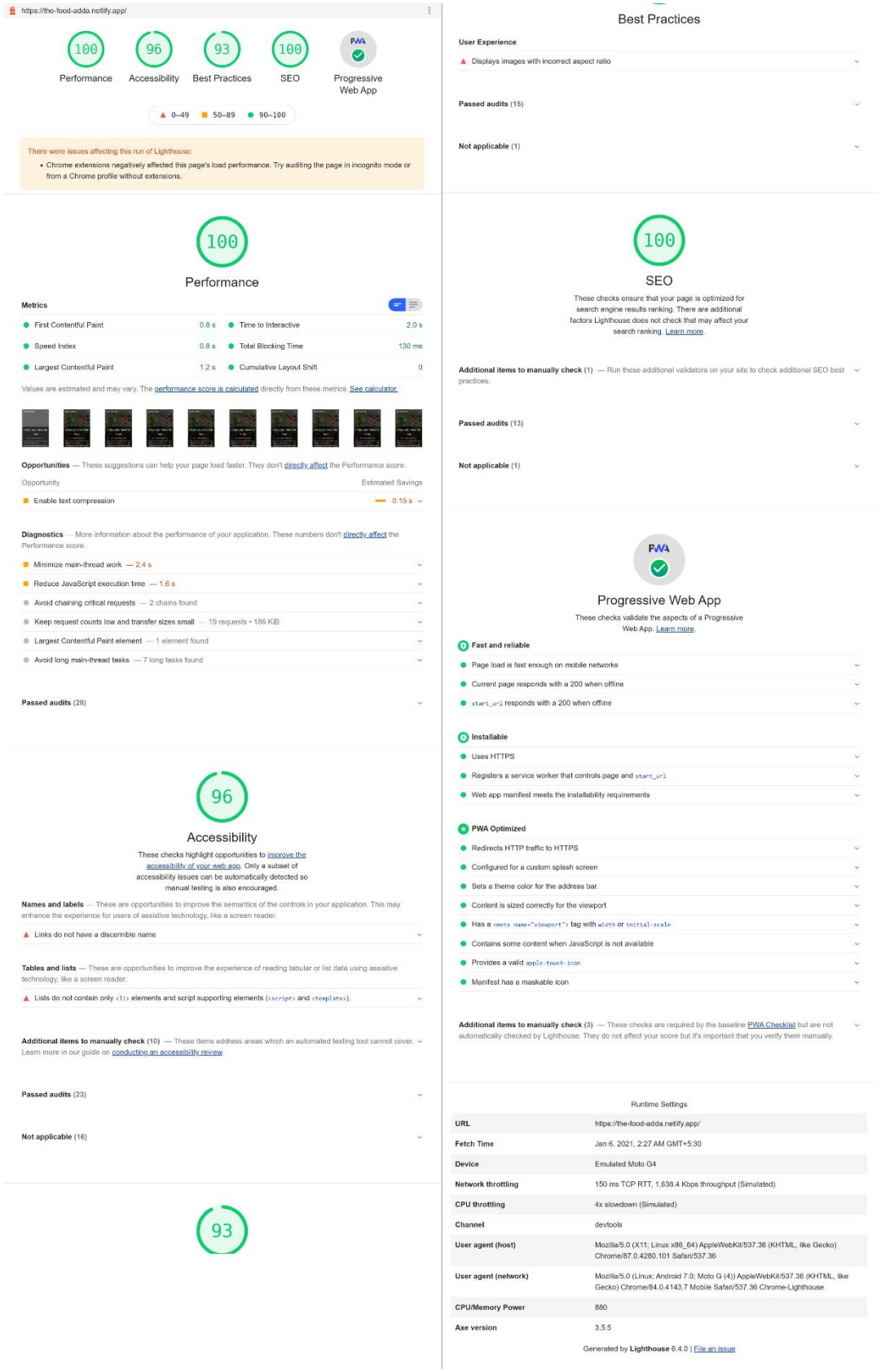


Fig. Report exported from Lighthouse

Coding Standard

We are using tools like prettier and ESLint to enforce *coding conventions and naming standards*, so that the code is maintainable, consistent, understandable, and flexible.

ESLint performs automated scans of JavaScript files for common syntax and style errors.

Prettier scans files for style issues and automatically reformats code to ensure consistent rules are being followed for indentation, spacing, semicolons, single quotes vs double quotes, etc.

We use them on project because:

- They keep everybody on the same page, following the same rules.
- They save time in code reviews, because we can safely ignore all style issues, and focus on things that matter, like the structure and semantics of our code.
- They catch errors. Prettier, not so much, but ESLint catches a lot of syntax errors and simple forms of type errors, such as undefined variables.
- Setting these things up is a one-time cost, but the time-saving benefits compound over time.

For files, naming conventions are used as well.

Limitations and Future Enhancements

This application has so far been made exclusively by focusing on orders, in which there is no payment option, which we will put later, when a real shopkeeper will buy and use it.

Currently, the consumer will have to pay in cash or otherwise, but in future they will have option to pay through website also.

We can add wallet for customers, and it will serve as prepaid.

Currently, we are storing images on same server where our backend is deployed.

We are going to use Amazon S3 bucket afterwards.

References and Bibliography

Almost everything in this documentation is followed by knowledge and wisdom of our guide and teachers. Beyond of this, we took care of notes from the following websites.

For learning,

React - https://reactjs.org

Material UI - https://material-ui.com

Node JS - https://nodejs.org

Mongo DB - https://docs.mongodb.com/manual/tutorial

Express - https://expressjs.com, and https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

Easy-peasy Library - https://easy-peasy.now.sh/docs/tutorials

JSON Web token - https://jwt.io/

Axios - https://www.npmjs.com/package/axios

Local Storage: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API

Moment JS - https://momentjs.com/docs

Bcryptjs - https://www.npmjs.com/package/bcryptjs

Gravatar - https://www.npmjs.com/package/gravatar

Passport - http://www.passportjs.org/docs

Validator - https://www.npmjs.com/package/validator

Safe-Regex - https://www.npmjs.com/package/safe-regex

Nodemon - https://www.npmjs.com/package/safe-regex

For dependencies, being used in system:

NPM JS: npm | build amazing things (npmjs.com)

Conclusion

This project boosted our knowledge and skills required in a corporate world specifically. Also, it exposed us to many new technologies and opportunities. We learnt teamwork, planning and human resource management.

Moreover, we are the only batch (hopefully) who got affected by a pandemic and completed whole project without any team members physical interaction, therefore it was a kind of challenge for every one of us but we learnt to overcome these limitations and achieve our goal.