Digital Menu Cards for Restaurants and Ordering System Using QR Code Scanning

PROJECT REPORT

submitted by

ARUN PRAKASH (Reg. No : PJR16CS015)

ATHIRA MANIYAN (Reg. No : PJR16CS017)

HASHIM M. (Reg. No : PJR16CS020)

to

APJ Abdul Kalam Technological University

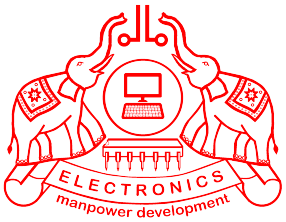
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering



Department of Computer Science and Engineering

College of Engineering Poonjar

Poonjar Thekkekkara P O, Kerala, India 686582

MAY 2020

DECLARATION

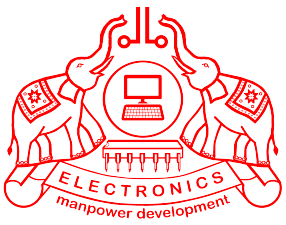
I undersigned hereby declare that the project report "**Digital Menu Cards for Restaurants and Ordering System Using QR Code Scanning**", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of **Mrs. Saira Shamsudeen**, Assistant Professor, Dept. of Computer Science & Engineering, College of Engineering Poonjar. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Poonjar
6/05/2020

Arun Prakash
Athira Maniyan
Hashim M.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING POONJAR



CERTIFICATE

This is to certify that the report entitled **Digital Menu Cards for Restaurants and Ordering System Using QR Code Scanning** submitted by **Ms. Athira Maniyan (PJR16CS017)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Saira Shamsudeen
Project Guide
Asst. Prof. of Computer Science
College of Engineering Poonjar

Mr. Rajesh K R
Head of Department
Computer Science
College of Engineering Poonjar

CONTENTS

ACKNOWLEDGEMENT

It is a great pleasure to acknowledge all those who have assisted and supported me for successfully completing my project.

First of all, I thank God Almighty for his blessings as it is only through his grace that I was able to complete my project successfully.

I take this opportunity to extend my sincere thanks to my project guide **Mrs. Saira Shamsudeen**, Assistant Professor, Dept. of Computer Science, College of Engineering Poonjar for her constant support and immense contribution for the success of my project.

I also extend my sincere thanks to all other members of the Department of Computer Science & Engineering for sharing their valuable comments during the preparation of the project.

I am also grateful to **Prof. Rajesh K. R.**, Project Coordinator and Head of Computer Science & Engineering Department, for the valuable guidance as well as timely advice which helped me a lot during the preparation of the project.

I am deeply indebted to **Dr. E S Jayachandran**, Principal, College of Engineering Poonjar for his encouragement and support.

I whole - heartedly thank all my classmates, for their valuable suggestions and for the spirit of healthy competition that existed between us.

ABSTRACT

An entirely new concept in the food industry is being introduced through this project. Our project "**Digital Menu Cards for Restaurants and Ordering System Using QR Code Scanning**" is an entirely new system to handle the ordering systems inside an hotel or restaurants. This project offers an efficient ordering system for the hotels and restaurants. Our Online Restaurants Menu Card System helps you to Streamline your operations to meet customers expectations. A flexible restaurant manage system which helps you to be more systematic and quick in your services to customers. Its fast and time management system for customers and as well as the hotel management.

The entier project is develpoed using Django and deployed in heroku. The storage space required while deploying the project is acquired using Amazon Web services. Some features of the project has been developed using java. The QR Code and Add-Ons are the applications of java. The proposed system has following modules incroporated in it. They are User Module, Food Item Management Module, Category Module, Confirm Order Module, Payment Module, Comment Module and Rating Module

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With increasing population, there has been a drastic increase in their demands and everyone is running out of time. The fact of significance is that no one wants to waste their time in long queue to get their task done . Usually one has to wait for a very long time to get their job done may be because of multiple load on the system receiving and processing the task or due to first-come first-serve approach which hinders those who have higher priority task to be done first but late to put request for execution. This problem is faced by a lot of organizations that still have the primitive processing system and also at a lot of places that are still dependent on manual work for processing requests.

There has not been much of advanced concept introduced in a lot of sectors that have a really good chance to bloom if they have the help of advanced technologies to attract more people and attention and hence a lot more profits. Since technology is always advancing, we can deploy it in real time applications to maximize the performance of the service systems and make it optimal.

The connectivity between the systems have not seen much advancement either when implemented in real world, the information transfers have always been happening in the old fashioned manner involving a lot of congestion and attenuation. There are chances that the request may enter in a deadlock situation if there is a collision of request initiating from the two or more consumers, and hence failing the system's throughput. The different sites that have installed a good communication system are facing the problem of network congestion, involving a lot of requests at the same time and no proper algorithm to regulate the requests and hence the quality of service deteriorates eventually causing packet loss, request delay and also blocking of new connections

There are some places where manual system is still considered a good option over automated systems for better interaction with the user and proper understanding of the situation, but the time factor is often ignored in such cases. There are areas that require immediate transfer of information and hence manual system is not a good choice. Also these areas are reluctant to adopt the new system due to high cost of the existing systems and hence they haven't implemented much advancement in their existing systems. On one side where manual systems are slow and there always is a scope for human errors, there lies automated systems on the other side that are fast, more informative for the users and also keeps record of all the processes and hence there is a very less scope for errors.

This Digital Menu Card System has been developed to override the problems prevailing in the manual orderin systems. This software eliminates and reduces the hardship faced in the exi manual systems.Moreover this is developed particularly for smooth and efficient functioning of the administration.

This application possibly reduces the error of serving wrong orders to the customers. No formal knowlegde is required to use this software. It is a user-friendly. The main objective of this application is an efficient management of deatils of Food item, cart, category, order, customer. It manages all the informations securely. the project is totaly built on administrator's end and thus only administrator is gauranted access.

CHAPTER 2

LITERATURE SURVEY

2.1 Online Food Ordering System - International Journal of Recent Technology and Engineering, Trupthi B, Rakshitha Raj R, J B Akshaya, Srilaxmi C P

An Online Food Ordering System is proposed here which simplifies the food ordering process. The proposed system shows an user interface and update the menu with all available options so that it eases the customer work. Customer can choose more than one item to make an order and can view order details before logging off. The order confirmation is sent to the customer. The order is placed in the queue and updated in the database and returned in real time. This system assists the staff to go through the orders in real time and process it efficiently with minimal errors.

2.2 Network Architecture for an Automated Service System

Present Service Systems are very primitive and time consuming where the customer has to first stand in the long queue to place their order and then have to wait to receive the order with uncertainty of amount of time to be spent on processing their order/task. This paper proposes an autonnted system that will enable customers to place the orders at public service systems in a timesaving manner; the custolller just needs to take the end handheld device which can be any tablet or phone and order either online from home, etc. making use of the central repository or using the local server present in the premises. This will enable direct processing of the task and the payment can be done in an easy manner through the same

end device. The system will have one main server connecting all the peers assuming more than one branch of the central system and also the local server to provide services to the user sitting in the System The customer will now get the status of the task done as in how much is it processed and how much more time will it take, a digital menu to select the options and also option to pay from his card directly, hence saving a lot of time at every point.

2.3 A Study on E-Commerce Applying in Taiwans Restaurant Franchise

When Taiwans traditional food industry facing the economic recession in 2008, except strengthen the constitution of themselves, they also promote the market scope to chain to develop franchise system aggressively. In order to promote the competition of enterprise and reduce management cost, this study that the writer counseling LUNGHSIENCHU will show the proven process of e-commerce, it becomes e- commerce little by little including retail store management, purchasing operation, service process, customer relationship management and promote management efficiency effectively. Then, they can enlarge franchise chain, food chain e-commerce and become operation model that cant be held back.

2.4 Research on E-commerce Mode of Food Enterprises

Food enterprises integrate supply chain management and e-commerce, in order to form complete e-supply chain. It is a better mode of promoting food businesses to develop rapidly. Food enterprises are rather unfamiliar with use of e-commerce for food enterprises, starting with the analysis of four stages of developinf e-commerce. With the analysis of advantages and deficiences of food enterprises developing e-commerce , the article discusses the objective requirementof food enterprises and proposes the counter measures based on the prsent situation of the food insudtry.

CHAPTER 3

SYSTEM DESIGN

3.1 DISADVANTAGES OF THE EXISTING SYSTEM

- Time consuming process.
- Need of paper for printing menu cards.
- Need of more number of man power.
- More amount of manual work to be accomplished.
- Chances of serving wrong dishes to cutsomers.
- Lacks customer satisfaction.
- Daily Updates is not possible through printed menu cards.

3.2 PROPOSED SYSTEM

The proposed system is an Digital Menu Card for ordering by scanning the QR Code. The system helps in elimination or the reduction of the manual labour and reduction of money spent on the man power. The proposed system helps the administration to operate and manage their resources smoothly and efficiently.

3.2.1 Advantages of Proposed System

- Decreases the load of the person involve in the existing manual system.
- Accuracy in work.
- Work becomes very speedy.

- Easy to update information.
- It contain better storage capacity.
- It tracks all the information of category, orders etc.
- Manages and integrates all the informations of the customers.
- Editing, adding and deleting of food items is easy and cost free.

3.3 SYSTEM ARCHITECTURE

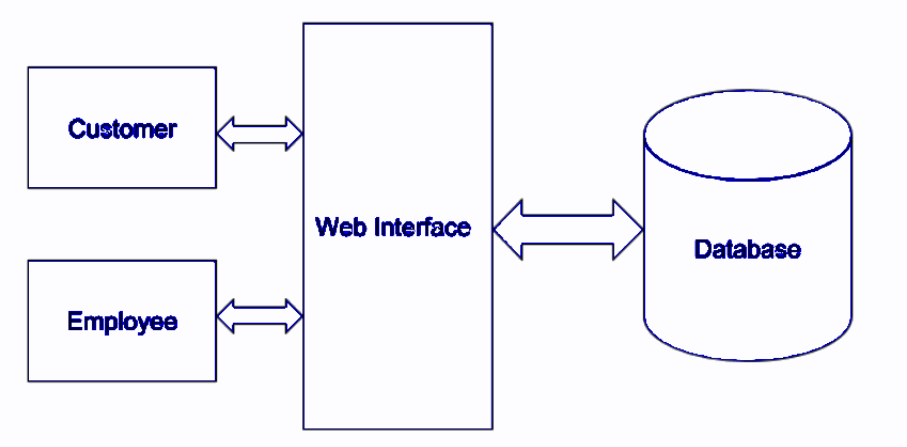


Figure 3.1: System Architecture

3.3.1 User Module

The main aim of the module is to provide all the functionalities related to the users. It tracks all the informations and details about the users. We have provided all the create, read, update and delete functionalities to this module. This is a role based module where admin can perform each and every operation on user data but users can only view his/her data. Thus access level restrictions has been implemented on the project. The admin can view the last login time of the users also.

Features :

- Customer will be able to see his details.
- Customer will be able to update his details.
- All customer forms are validated on client side using Django.

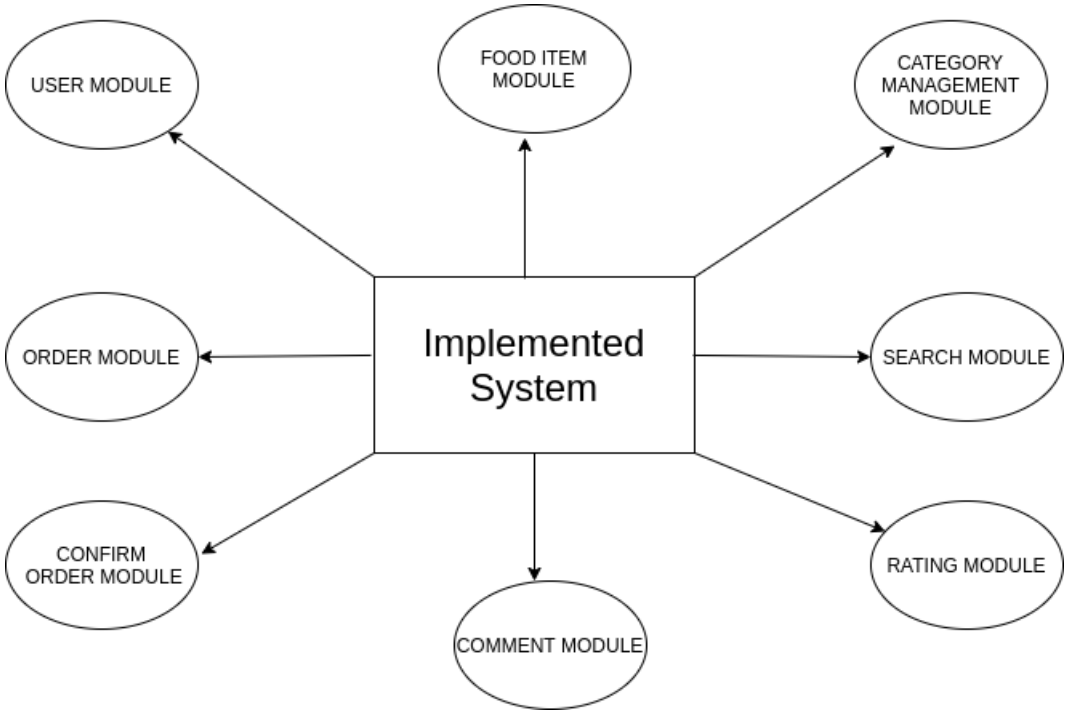


Figure 3.2: Modules Of the System

Field	Type
Email_Id	Primay Key
First_Name	Varchar
Last_Name	Varchar

Table 3.1: User Table

- Only admin can edit and update the record of the customer.
- Admin can add new customer.
- Admin can see the list of customer details.
- Admin will be able to delete the records of the customer.

3.3.2 Food Item Management Module

The main aim of developing this module is to manage all the deltails of the food items being served in the restaurant. So all the records of the food items can be update by the admin and the customers can view the food item lists. Also the module manages the order detail i.e the food items ordered and the details of the table no. where that food item has to be served. The customers can view the making vedio of the food item they want to order and also the daily offers are also displayed.

Field	Type
User_Id	Primay Key
Item	Varchar
Quantity	Integer

Table 3.2: Cart Table

Field	Type
User_Id	Primay Key
Ordered_Items	Varchar
Table_No	Integer

Table 3.3: Order Table

Features :

- Admin can manage the food.
- Admin can edit/delete the food.
- Admin can see the list of all food.
- Customer can see food.
- Customers can view their placed orders.
- Customers can select their desired food item from the list.

3.3.3 Category Management Module

This module manages the categories being served in the restaurant. It provides the customers an view to know the food categories available. This is categories the food items into drinks, lunch dinner, breakfast etc. This helps the customers to easily access to their favorite food item. The food items are categorised as per the admin needs. Admin can alter the category as they require.

Features :

- Admin can manage the item category.
- Admin can edit/delete the item category.
- Admin can see the list of all item category.
- Customer can see item category.

Field	Type
Title_Id	Primay Key
Primary_Category	Boolean

Table 3.4: Catrgory Table

Field	Type
Object_Id	Primay Key
Name	Varchar
Category	Integer
Detail	Varchar
Price	Varchar

Table 3.5: Product Table

3.3.4 Confirm Order Module

The main aim of this module is provide all the functionality realted to confirm order. So all confirm order will be managed by admin and customer will be able to see confirm order. It tracks all the information and details of the confirm order.

Features :

- Admin can add new confirm order.
- Admin can see the list of confirm order details.
- Only admin can edit and update the record of the confirm order.
- Admin will be able to delete the records of the confirm order.
- All confirm order forms are validated on client side using Django.

3.3.5 Comment Module

Field	Type
User	Primay Key
Address	Varchar
ZipCode	Integer
Landmark	Varchar

Table 3.6: Billing address Table

Field	Type
Object_Id	Primay Key
Name	Varchar
Content Type	Varchar
Date-time	date

Table 3.7: Comment Table

The main aim of developing this module is to get the feedback of the food items from customers. The number of comment of each food item is displayed to the customers. All the comments are visible to the customers.

Features

- Costumers can comment for each food item on the list.
- Comment can be visible with time and date on the Food item management module.
- Admin can add or delete the comments.
- Total number of comments is mentioned.

We have developed this comments using django comments. Django includes a comments framework. This framework can be used to attach comments to any model, so you can use it for comments on blog entries, photos, book chapters, or anything else.

A Roadmap of how we built this comment system

- Created a model to save the comments.
- Created a form to submit comments and validate the input data.
- Added a view that processes the form and saves the new comment to the database.
- Edited the post detail template to display the list of comments and the form to add a new comment.

3.3.6 QR Code scanning

In our proposed system we provide a QR Code on every table whih will be regenerated after every 30 seconds. Scanning the QR Code the customers will be taken to our proposed website from where they can access our menu card to place their desiered order.

Field	Type
Object_Id	Primay Key
Count	Integer
Total	Integer
Average	Integer

Table 3.8: Rating Table

Field	Type
Object_Id	Primay Key
User	Varchar
Score	Integer
Rating	Varchar

Table 3.9: User Rating Table

3.3.7 Rating Module

The main aim of this module to allow the users and admin to rate each food item. The rating of a food item is taken as average ratings of all the customers. Admin cannot alter the ratings, the rating is dispalyed as the average rating of the all for each food item.

Features :

- Admin cannot change the rating as per their needs.
- We have developed an very genuine rating system.
- Average ratings are showcased fro each food item.

3.3.8 Payment Module

The main purpose for developing this module is to manage the customer's payment via online or cash on delivery. Admin will manage all cash on delivery record. As per the customer convenience, customer can pay their bills. Admin manages all the payment records of all the customers.

CHAPTER 4

IMPLEMENTATION

4.1 OVERVIEW

In this proposed system, a QR Code is regenertated for every 30 seconds. This protects the uniqueness of this project. The customers can scan the QR Code in order to access the website we have developed. The customers can pick their desired Food items from the categorised food item list. And drop into their carts. They can place the orders and pay their bills after confirming their orders. the payment can be done through the payment module or can pay on desk after the satisfactory meal. This can be done according to the customers convenience.

4.2 Implemenetation of the Modules

The proposed system is majorly devepoled using the Django Framework. Django is a Python-based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a 501(c)(3) non-profit.

Django is a Python-based free and open-source web framework, which follows the model-template-view architectural pattern. It is maintained by the Django Software Foundation, an independent organization established as a 501 non-profit. Django's primary goal is to ease the creation of complex, database-driven websites.

Despite having its own nomenclature, such as naming the callable objects generating the

HTTP responses ”views”,[5] the core Django framework can be seen as an MVC architecture.[6] It consists of an object-relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database (”Model”), a system for processing HTTP requests with a web templating system (”View”), and a regular-expression-based URL dispatcher (”Controller”).

Also included in the core framework are:

- a lightweight and standalone web server for development and testing
- a form serialization and validation system that can translate between HTML forms and values suitable for storage in the database
- a template system that utilizes the concept of inheritance borrowed from object-oriented programming
- a caching framework that can use any of several cache methods
- an interface to Python’s built-in unit test framework
- Django REST framework is a powerful and flexible toolkit for building Web APIs.

The basic layout of the website has been developed using HTML, CSS and bootstrap. Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

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

4.3 Implementation of Payment Module

A Payment System is a mechanism that facilitates transfer of value between a payer and a beneficiary by which the payer discharges the payment obligations to the beneficiary. Payment Systems are the medium to transfer funds from one person to another that facilitate businesses and economies.

The main purpose for developing this module is to manage the customer's payment via online or cash on delivery. Admin will manage all cash on delivery record. As per the customer convenience, customer can pay their bills. Admin manages all the payment records of all the customers.

We have developed the payement module using Stripe API. There are many payment services available in the market to integrate payment gateway in an application. For example, PayPal, Stripe, Sage Pay, CCAvenue and there is a long list out there. They provide API for integrating payment gateway to our software. Here we have used Stripe Payment Gateway Integration using Django. In many countries, Stripe is the widely used for the transactions with credit and debit cards. By using a payment gateway services / API, we can enable users to do financial transactions with our application. When it comes to integrating a payment gateway with our application, we need to choose a reputed provider. Because it gives trust to the users and it is important as it involves real money.

We can use the Stripe API in test mode, which does not affect your live data or interact with the banking networks. The API key we use to authenticate the request determines whether the request is live mode or test mode. The Stripe API differs for every account as it releases new versions and tailor functionality.

Errors : Stripe uses conventional HTTP response codes to indicate the success or failure of an API request. In general: Codes in the 2xx range indicate success. Codes in the 4xx range indicate an error that failed given the information provided (e.g., a required parameter was omitted, a charge failed, etc.). Codes in the 5xx range indicate an error with Stripe's servers (these are rare).

Handling Errors : The Client libraries available raise exceptions for many reasons, such

Table 4.1: HTTP STATUS CODE SUMMARY

200-OK	Everything worked as expected.
400-Bad Request	The request was unacceptable, often due to missing a required parameter.
401-Unauthorized	No valid API key provided.
402-Request Failed	The parameters are valid but the request failed.
403-Forbidden	The API key's does not have the permission to perform the request.
404-Not Found	The requested resource dose not found.
409-Conflict	The request conflicts with another request.
429-Too Many Requests	Too many requests hit the API quickly.
500, 502, 503, 504	Something went wrong on stripe's end.

Table 4.2: ERROR TYPES

api_connection_error	Failure to connect stripe's api.
api_error	API errors cover any other type problem and are extremely uncommon.
authentication_error	Failure to properly authenticate yourself in the request.
card-error	most common type of error occurs user enters a card that can be charged.
rate_limit_error	Too many requests hit the API quickly.
invalid_request_error	This error arises when the request has invalid parameters.

as a failed charge, invalid parameters, authentication errors, and network unavailability. It recommends writing code that gracefully handles all possible API exceptions.

Idempotent Requests : The API supports idempotency for safely retrying requests without accidentally performing the same operation twice. This is useful when an API call is disrupted in transit and you do not receive a response. For example, if a request to create a charge does not respond due to a network connection error, you can retry the request with the same idempotency key to guarantee that no more than one charge is created. To perform an idempotent request, provide an additional idempotency_key element to the request options. Stripe's idempotency works by saving the resulting status code and body of the first request made for any given idempotency key, regardless of whether it succeeded or failed. Subsequent requests with the same key return the same result, including 500 errors.

An idempotency key is a unique value generated by the client which the server uses to recognize subsequent retries of the same request. How you create unique keys is up to you, but we suggest using V4 UUIDs, or another random string with enough entropy to avoid collisions. Keys are eligible to be removed from the system after they're at least 24 hours old, and a new request is generated if a key is reused after the original has been pruned. The idempotency layer compares incoming parameters to those of the original request and errors

unless they're the same to prevent accidental misuse. Results are only saved if an API endpoint started executing. If incoming parameters failed validation, or the request conflicted with another that was executing concurrently, no idempotent result is saved because no API endpoint began execution. It is safe to retry these requests. All POST requests accept idempotency keys. Sending idempotency keys in GET and DELETE requests has no effect and should be avoided, as these requests are idempotent by definition.

4.4 Implementation of Social Authentication

Adding Social Authentication to Django. Python Social Auth is a library that provides an easy-to-setup social authentication / registration mechanism with support for several frameworks and auth providers. Django comes with a user authentication system. It handles user accounts, groups, permissions and cookie-based user sessions. The Django authentication system handles both authentication and authorization. Briefly, authentication verifies a user is who they claim to be, and authorization determines what an authenticated user is allowed to do. Here the term authentication is used to refer to both tasks.

The auth system consists of:

- Users
- Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
- Groups: A generic way of applying labels and permissions to more than one user.
- A configurable password hashing system
- Forms and view tools for logging in users, or restricting content
- A pluggable backend system

Python Social Auth aims to be an easytosetup social authentication and authorization mechanism for Python projects supporting protocols like OAuth (1 and 2), OpenID and others. The initial codebase is derived from django-social-auth with the idea of generalizing the process to suit the different frameworks around, providing the needed tools to bring support to new frameworks. django-social-auth itself was a product of modified code from django-twitter-oauth and django-openid-auth projects.

4.5 Implementation of Django Filters

Django-filter is a generic, reusable application to alleviate writing some of the more mundane bits of view code. Specifically, it allows users to filter down a queryset based on a model's fields, displaying the form. Django-filter provides a simple way to filter down a queryset based on parameters a user provides.

4.6 Implementation of QR Code Scanning

In our proposed system, QR code Scanning is the first step to access our web page. By simply scanning the provided Qr code. After which the customers can accesses our web page easily. There is no need to download any application form the play store and also no need to search for our website over the internet. To maintain the uniqueness of the code the QR Code is regenerated every 30 seconds.

A QR Code is a machine-readable code consisting of an array of black and white squares, typically used for storing URLs or other information for reading by the camera on a smartphone.A QR code (short for "quick response" code) is a type of barcode that contains a matrix of dots. It can be scanned using a QR scanner or a smartphone with built-in camera. Basically, a QR code works in the same way as a barcode at the supermarket. It is a machine-scannable image that can instantly be read using a Smartphone camera. Every QR code consists of a number of black squares and dots which represent certain pieces of information.

All QR codes have a square shape and include three square outlines in the bottom-left, top-left, and top-right corners. These square outlines define the orientation of the code. The dots within the QR code contain format and version information as well as the content itself. QR codes also include a certain level of error correction, defined as L, M, Q, or H. A low amount of error correction (L) allows the QR code to contain more content, while higher error correction (H) makes the code easier to scan.

QR codes have two significant benefits over traditional UPCs the barcodes commonly used in retail packaging. First, since QR codes are two-dimensional, they can contain signif-

icantly more data than a one-dimensional UPC. While a UPC may include up to 25 different characters, a 33x33 (version 4) QR code, can contain 640 bits or 114 alphanumeric characters. A 177x177 (version 40) QR code can store up to 23,648 bits or 4,296 characters.

Another advantage of QR codes is that they can be scanned from a screen. Standard UPC scanners use a laser to scan barcodes, which means they typically cannot scan a UPC from a screen (like a smartphone). QR scanners, however, are designed to capture 2D images printed on paper or displayed on a screen. This makes it possible to use a QR code on your smartphone as a boarding pass at the airport or as a ticket for an event.

This feautre is implemented using JavaScript and Jquery. The QR code of the proposed system and the add-ons are developed using JavaScript. JavaScript s a programming language that conforms to the ECMAScript specification.[7] JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

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

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

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

jQuery also provides a paradigm for event handling that goes beyond basic DOM element selection and manipulation. The event assignment and the event callback function definition are done in a single step in a single location in the code. jQuery also aims to incorporate other highly used JavaScript functionality (e.g. fade ins and fade outs when hiding elements, animations by manipulating CSS properties). The jQuery library is typically distributed as a single JavaScript file that defines all its interfaces, including DOM, Events, and Ajax functions. It can be included within a Web page by linking to a local copy, or by linking to one of the many copies available from public servers. jQuery has a content delivery network (CDN) hosted by MaxCDN.

4.7 Creating Databases

In the proposed system the databases has been created using ProstgreSQL. PostgreSQL is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. t is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users. It is the default database for macOS Server.

PostgreSQL manages concurrency through multiversion concurrency control (MVCC), which gives each transaction a "snapshot" of the database, allowing changes to be made without affecting other transactions. This largely eliminates the need for read locks, and ensures the database maintains ACID principles. PostgreSQL offers three levels of transaction isolation: Read Committed, Repeatable Read and Serializable. Because PostgreSQL is immune to dirty reads, requesting a Read Uncommitted transaction isolation level provides read committed instead. PostgreSQL supports full serializability via the serializable snapshot

isolation (SSI) method.

In PostgreSQL, a schema holds all objects, except for roles and tablespaces. Schemas effectively act like namespaces, allowing objects of the same name to co-exist in the same database. By default, newly created databases have a schema called public, but any further schemas can be added, and the public schema isn't mandatory. A search_path setting determines the order in which PostgreSQL checks schemas for unqualified objects (those without a prefixed schema). By default, it is set to $user, public ($user refers to the currently connected database user). This default can be set on a database or role level, but as it is a session parameter, it can be freely changed (even multiple times) during a client session, affecting that session only.

4.8 Deploying the Proposed System

The proposed system is depoyed on heroku. Heroku is a cloud platform as a service (PaaS) supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it has features for a developer to build, run and scale applications in a similar manner across most languages.

Developers use Heroku to deploy, manage, and scale modern apps. The platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market. Heroku is fully managed, giving developers the freedom to focus on their core product without the distraction of maintaining servers, hardware, or infrastructure. The Heroku experience provides services, tools, workflows, and polyglot supportall designed to enhance developer productivity.

Heroku's free cloud services begins with the apps - apps which can be deployed to dynos - the lightweight Linux containers that are at the heart of the Heroku platform. When the user sign up with Heroku, you automatically get a pool of free dyno hours to use for the apps lications. When the application runs, it consumes dyno hours.

Heroku gives a set of powerful capabilities that deliver higher-order value. The Heroku platform is fully-managed, meaning that it takes care of servers, hardware, and infrastructure, so that it can stay focused on the application. The platforms flexibility allows to build apps using the preferred language or framework, and using popular architectural patterns, such as microservices. Deploying apps on Heroku is fast and streamlined, with built-in workflows that support continuous integration and continuous delivery practices. The platforms operational experience offers built-in tools for easily scaling and maintaining application health. Many Heroku developers use a range of free services to experiment, learn, and test out new ideas with their users before they are ready to scale.

The required storage space while deploying the enitre project was acquired by using the Amazon Web Services (AWS). Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis. In aggregate, these cloud computing web services provide a set of primitive abstract technical infrastructure and distributed computing building blocks and tools. One of these services is Amazon Elastic Compute Cloud, which allows users to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer, including hardware central processing units (CPUs) and graphics processing units (GPUs) for processing; local/RAM memory; hard-disk/SSD storage; a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, and customer relationship management (CRM).

Amazon Simple Storage Service (S3) is a highly durable and available store that is ideal for storing application content such as media files, static assets, and user uploads. S3 allows you to offload the storage of static files from your app. This is crucial on Heroku, because your apps dynos have an ephemeral filesystem. This means that all files that arent part of your applications slug are lost whenever a dyno restarts or is replaced (this happens at least once daily).

All files in S3 are stored in buckets. Buckets act as a top-level container, much like a directory. All files sent to S3 belong to a bucket, and a buckets name must be unique across all of S3. Access to the S3 API is governed by an Access Key ID and a Secret Access Key. The access key identifies your S3 user account, and the secret key is a password-like credential that should be stored securely.

CHAPTER 5

RESULTS

The advancement in data and communication technology has greatly influenced the business transactions. In earlier days, food industry has lagged behind alternative industries in adopting new technology. But speedy advances in technology and heightened expectations of customers and have forced the food industry to bring new innovative methods. Nowadays, the adoption of wireless technology and emergence of mobile devices has a big role within the food industry. The business and services in restaurants are often improved with the mixture of wireless and mobile technologies. The competition in restaurants with regard to business has redoubled with the advancements in food ordering techniques. In this paper, a digital menu card for restaurants and ordering system using QR code scanning is proposed which is able to streamline the operations to meet customers expectations. A flexible restaurant manage system which helps you to be more systematic and quick in your services to customers. The implementation of proposed application uses HTML/CSS and Java as the front end and at the backend Python, Django Framework along with MySQL database is used.

CHAPTER 6

FUTURE SCOPE

In current formal dining environments, some form of physical static menu is utilized to convey the available food and beverage choices to customers. Said menus are generally paper based and hence impose restrictions on the textual real estate available and the ability a restaurateur has to update them. This document specifies the requirements for a restaurant paper menu and ordering replacement strategy to alleviate the problems associated with the current archaic method.

The existing system is paper based. The traditional menu cards in the restaurants are paper based. Waiters use paper to write the order of customers. The records are stored on paper. As with anything paper based, it is so easy for things to get damaged by Coffee stains etc, or paper being lost due to fire or accidents or just generally lost. There is wastage of time, money, and paper. As traditional menu cards are paper based, any changes that need to be made in the menu card will lead to wastage. As it will require reprinting of all the menu cards. Also, for small changes it is not possible to print all the menu cards again and again. There is no power to dynamically make any changes in the menu card. To access a particular record from the stack of papers is not efficient. From the customers point of view, this system is time consuming. As, one has to wait until the waiter comes to take the order, one has to call waiter number of times till he notices it, there can be misinterpretation while the waiter is writing your order on paper, and it might be possible that you are served with a wrong dish.

There has been improvements in the management of restaurants. Each waiter is assigned a group of tables, after taking orders for a table the waiters enter the orders (a list of dishes and drinks ordered by the diner or group of diners) into the system at the PC. The waiter

usually knows of any dishes that are unavailable before taking an order. The system must confirm the availability of dishes. Should an item not be available the system must allow the waiter to change or even delete a customers order. Dishes to be prepared are sent to the kitchen, drinks orders to the bar. Starters and main course orders are usually taken together. Drinks and desert orders may be taken separately. Kitchen staff sees the dish orders on their screen, prepare them in an appropriate sequence and confirm preparation to the system when complete, similarly with the bar. When a waiter sees the completion indications on his terminal he collects the items and takes them to the table. The waiter can also check on the status of dish and drink orders. At the end of the meal the waiter will have the system print a bill, and he will enter the details of payment for it. The management can give discounts. The system keeps track of the numbers of customers served by each waiter and the amount of money taken by each waiter.

There are still many areas which are not closely looked at. Like, making dynamic changes in the menu card, to get rid away from the heap of paper based records, to assure the customer that hell be served with what he has ordered, to get the customer feedback on record.

CHAPTER 7

CONCLUSION

In this proposed system we have a QR Code in the menu card in each table of the restaurant. For more details about the food stuff one can just scan the QR Code for the details. After the QR Code scanning the customer gets into our website page displaying the menu items and their respective price. In this page one can see athe entier detail about the food stuff selected. The existing system of Hotel Management is manual. All the daily routines is carried out manually and the records are maintained in the record books or the registers. The introduction of our project will eliminate a good amount of mannual work. In this proposed restaurant menu card system, displays the food stuffs available there and their respective prices. Along with the displaying of food prices, the customer can also view the making of the foodstuffs. Not only the prices are displayed, but also the making of each food stuff is made available to the customers. In addition to these features we have an ordering system. Using this ordering system the customer can order their preferences without waiting for the waiters. This concept is satisfactory for both the hotel administration and the customers.

This paper has succeeded in designing a cost effective and less manual effort needed ordering system. The proposed system helps you to Streamline your operations to meet customers expectations. A flexible restaurant manage system which helps you to be more systematic and quick in your services to customers. Its fast and time management system for customers. The paper also verified the designed system by constructing and testing a prototype of the system using the components outlined in this paper.

REFERENCES

[1] Ward, Paul T. "The transformation schema: An extension of the data flow diagram to represent control and timing." Software Engineering, IEEE Transactions on 2 (1986): 198-210.

[2] Fowler, Remy J., and Will E. Leland. "Local area network characteristics, with implications for broadband network congestion management." Selected Areas in Communications, IEEE Journal on 9.7 (1991): 1139-1149.

[3] Wang, Zhikui, and Fernando Paganini. "Global stability with time-delay in network congestion control." Decision and ControL 2002, Proceedings of the 41stIEEE Conference on. Vol. 4. IEEE, 2002. [8] Berman, Keith, et al. "Automated networked serv

[4] Awojide, Simon, I. M. Omogbhemhe, O. S. Awe, and T. S. Babatope, Towards the digitalization of Restaurant Business Process for Food Ordering in Nigeria Private University: The Design Perspective. A Study of Samuel Adegboyega University Edo State Nigeria, Int. J. Sci. Res. Publ., vol. 8, no. 5, pp. 4654, 2018.

[5] Varsha Chavan, Priya Jadhav, Snehal Korade, Priyanka Teli, Implementing Customizable Online Food Ordering System Using Web Based Application, International Journal of Innovative Science, Engineering Technology(IJISET) 2015.

[6] Tennenhouse, David L., and David 1. Wetherall. "Towards an active network architecture." DARPA Active Networks Conference and Exposition, 2002.

[7] Schneider, Eric. "Method, product, and apparatus for processing a data request." U. S. Patent No.6,760,746. 6 Jul. 2004.

Appendices

Appendix A

Source Code

```
{% extends 'products/base.html' %}
{% load ratings %}

{% block content %}


<div class="container my-5">


<div class="card">
    <div class="row">
        <aside class="col-sm-5 border-right">
<article class="gallery-wrap">
<div class="img-big-wrap">
  <div> <a href="#"><img src="/media/{{ object.mainimage }}" style="widt
</div> <!-- slider-product.// -->
</article> <!-- gallery-wrap .end// -->
        </aside>
        <aside class="col-sm-7">
<article class="card-body p-6">
    <h3 class="title mb-3">{{ object.name }}</h3>
<p class="price-detail-wrap">
    <span class="price h3 text-warning">
        <span class="currency">   </span><span class="num">{{ object.pri
    </span>
</p> <!-- price-detail-wrap .// -->
<dl class="item-property">
```

```
  <dt>Description</dt>
  <dd><p>{{ object.preview_text }}</p></dd>

</dl>
{% ratings object %}
<a href="{% url 'mainapp:cart' object.slug %}" class="btn btn-lg btn-ou
</article> <!-- card-body.// -->
<br><br>
<h3 class="title mb-3">Making Video of {{ object.name }}</h3>

        <video width='400' controls>
        <source src='{{ MEDIA_URL }}{{ object.videofile }}' type='video/
        Your browser does not support the video tag.
        </video>
        <br><br>
         <!-- col.// -->
    </div> <!-- row.// -->
</aside>

    <br><br>

</div> <!-- card.// -->


</div>
<!--container.//-->

{% endblock %}

{% extends 'products/base.html' %} {% load crispy_forms_tags %}
 {% block content %}

<div class="container mt-5">
  <h2 class="mb-3">Le cafe ' > <span class="text-muted">Checkout</span><
  <div class="col-md-3">
    <div class="card" style="height: auto">
      <div class="card-body">
```

```
          <ul class="list-group list-group-flush">
            {% for item in order_items %}
            <li class="list-group-item">
              {{ item.item.name }} x {{ item.quantity }}
            </li>
            {% endfor %}
          </ul>
        </div>

        <div class="card-footer">
          <span class="float-left"><b>Order Total</b></span>
          <span class="float-right"
            ><b>    {{ order_total | floatformat:2 }}</b></span
          >
        </div>
      </div>
      <div class="card mb-5" style="height: auto">
        <div class="card-body">
          <a href=" {% url 'checkout:payment' %} " class="btn btn-primary
       </div>
      </div>
    </div>
</div>
{% endblock %}

{% extends 'products/base.html' %}
{% load crispy_forms_tags %}
{% block content %}

<div class="container mt-5">
<h2 class="text-center"><strong>Your Ordes</strong></h2>
<div class="row">
      <div class="col-md-12">
          <div class="card mt-5" style="height: auto">
              <div class="table-responsive">
              <table class="table">
                <thead>
                  <tr>
```

```
                    <th scope="col">#</th>
                    <th scope="col">Table_no</th>
                    <th scope="col">Products</th>
                    <th scope="col">Status</th>
                  </tr>
                </thead>
                <tbody>


                  {% for order in orders %}
                  <tr>
                    <th scope="row">{{ forloop.counter }}</th>
                    <td><a href="#">Table {{ order.table_no }}</a></td>
                    <td>
                      {% for item in order.orderitems.all %}
                          {{ item }}
                          <br>
                      {% endfor %}
                    </td>
                    <td><span class="badge badge-primary">Processing Your
                  </tr>
                    {% endfor %}

                </tbody>
              </table>
              </div>
          </div>
      </div>
      <div class="col-md-12 my-5 text-center">
          <a href="/" class="btn btn-success">Back to Home</a>
      </div>
</div>
</div>

{% endblock %}
```

payment.html

```
{% extends 'products/base.html' %} {% load crispy_forms_tags %}
{% block content %}

<div class="container">
  <div class="jumbotron">
    <h1>Choose your payment option</h1>
  </div>
</div>
<br />
<h1 class="text-center">Grand Total:      {{ tot }}</h1>
<h5 class="text-center">
  including all service charges. (no delivery charges applied)
</h5>
<br />
<div class="container mt-5 text-center">
  <h1 class="text-center">
    <a href="{% url 'mainapp:cart-home' %}"
    ><button class="btn btn-warning">
      <span class="glyphicon glyphicon-circle-arrow-left"></span> Go bac
      cart
    </button></a
  >
  <form action="{% url 'checkout:cashonpay' %}" method="post">
    {% csrf_token %}

  <a
  ><button
    type="submit"
    name="success"
    value="success"
    class="btn btn-success"
  >
    <span class="glyphicon glyphicon-"></span> Cash On Delivery
  </button></a
>

</form>
</h1>
```

```
  <form action="{% url 'checkout:charge' %}" method="POST">
    {% csrf_token %}
   <script
      src="https://checkout.stripe.com/checkout.js"
      class="stripe-button"
      data-key="{{ key }}"
      data-description="Complete your order"
      data-amount="{{ total }}"
      data-locale="auto"
    ></script>
  </form>
</div>

{% endblock %}

table.html

{% extends 'products/base.html' %}
 {% block content %}

<div class="container mt-5">
    <div class="col-md-3">
      <div class="card" style="height: auto">
    <div class="card-body">
      <form method="POST" action="{% url 'checkout:table' %}" >
        {% csrf_token %}
        <label>

          <span class="float-left"><b>Enter Your Table No :</b></span>
          <input
            type="number"
            id="table"
            name="table"
            min="1"
            max="20"
          />
        </label>
```

```
            <br />
            <button type="submit" class="btn btn-success">
              submit table_number
            </button>
          </form>
        </div>

        </div>
        <div class="card-body">
          <a href=" {% url 'checkout:index' %} " class="btn btn-primary floa
        </div>
        </div>

        </div>

{% endblock %}
```

home.html

```
{% extends 'products/base.html' %}
{% load ratings %}
{% block content %}

<div class="container">
    <!--Slideshow starts here -->
    {% for product, range, slideNo in prodArr %}
    <h5 class="my-4">{{product.0.category}} - Recommended Items</h5>
    <div class="row">
        <div id="demo{{forloop.counter}}" class="col carousel slide my-
            <ul class="carousel-indicators">
                <li data-target="#demo{{forloop.counter}}" data-slide-to
                {% for i in range %}
                <li data-target="#demo{{forloop.parentloop.counter}}" da
                {% endfor %}
            </ul>
            <div class="container carousel-inner no-padding">
```

```
                    <div class="carousel-item active">
                        {% for i in product %}
                        <div class="col-xs-3 col-sm-3 col-md-3">
                            <div class="card-deck">
                                <div class="card align-items-center" style="
                                    <img src='/media/{{i.mainimage}}' class=
                                    <div class="card-body">
                                        <h5 class="card-title" id="namepr{{i
                                        <h6 class="card-title" >  <span id=
                                    </div>
                                </div>
                            </div>
                        </div>
                                        {% if forloop.counter|divisibleb
                                            </div>
            <div class="carousel-item">
                {% endif %} {% endfor %}
            </div>
        </div>
    </div>
        <!-- left and right controls for the slide -->
            <a class="carousel-control-prev" href="#demo{{forloop.counter}}"
        <span class="carousel-control-prev-icon"></span>
        </a>
            <a class="carousel-control-next" href="#demo{{forloop.counter}}"
        <span class="carousel-control-next-icon"></span>
        </a>
</div>
{% endfor %}
</div>


{% endblock %}
```

home.html

```
{% extends 'products/base.html' %}
```

```
{% load ratings %}
{% block content %}

<div class="container">
    <!--Slideshow starts here -->
    {% for product, range, slideNo in prodArr %}
    <h5 class="my-4">{{product.0.category}} - Recommended Items</h5>
    <div class="row">
        <div id="demo{{forloop.counter}}" class="col carousel slide my-
            <ul class="carousel-indicators">
                <li data-target="#demo{{forloop.counter}}" data-slide-to
                {% for i in range %}
                <li data-target="#demo{{forloop.parentloop.counter}}" da
                {% endfor %}
            </ul>
            <div class="container carousel-inner no-padding">
                <div class="carousel-item active">
                    {% for i in product %}
                    <div class="col-xs-3 col-sm-3 col-md-3">
                        <div class="card-deck">
                            <div class="card align-items-center" style="
                                <img src='/media/{{i.mainimage}}' class=
                                <div class="card-body">
                                    <h5 class="card-title" id="namepr{{i
                                    <h6 class="card-title" >  <span id=
                                </div>
                            </div>
                        </div>
                    </div>
                    {% if forloop.counter|divisibleby:4 and forloop.cou
                </div>
                <div class="carousel-item">
                    {% endif %} {% endfor %}
                </div>
            </div>
        </div>
        <!-- left and right controls for the slide -->
        <a class="carousel-control-prev" href="#demo{{forloop.counter}}"
```

```
            <span class="carousel-control-prev-icon"></span>
        </a>
        <a class="carousel-control-next" href="#demo{{forloop.counter}}"
            <span class="carousel-control-next-icon"></span>
        </a>
    </div>
    {% endfor %}
</div>


{% endblock %}
```

Appendix B

Screenshots

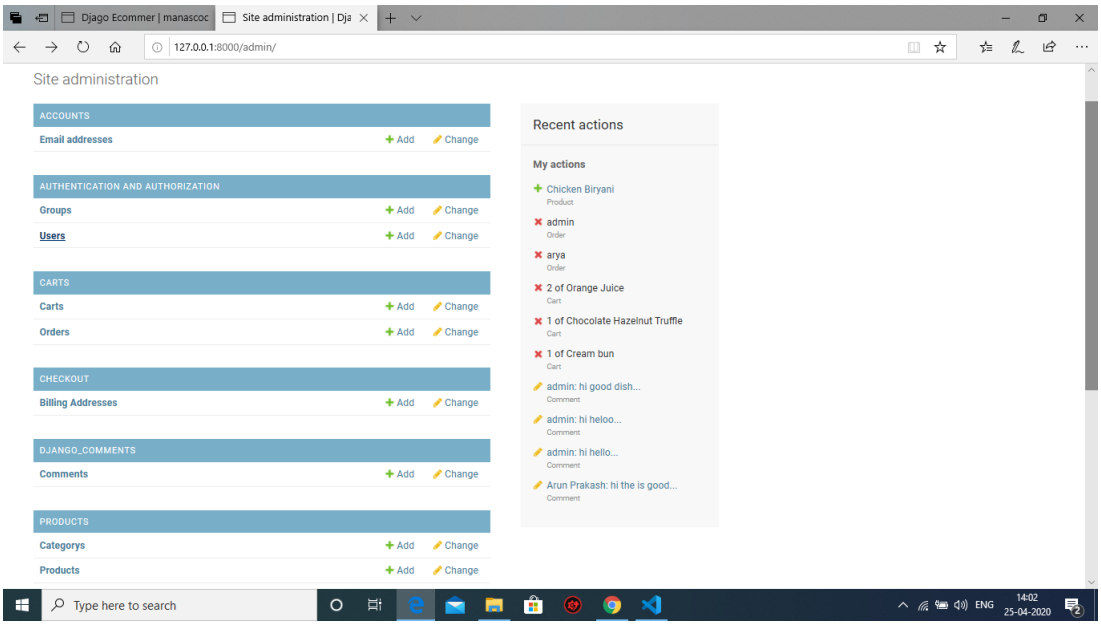


Figure B.1: System Modules

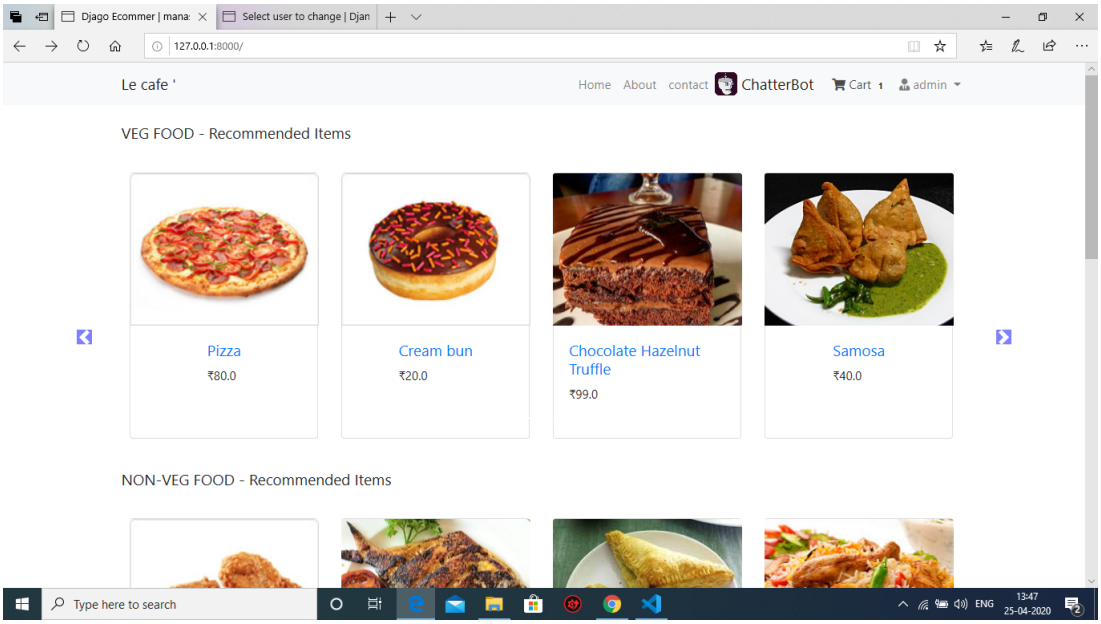


Figure B.2: Food Item Management Module

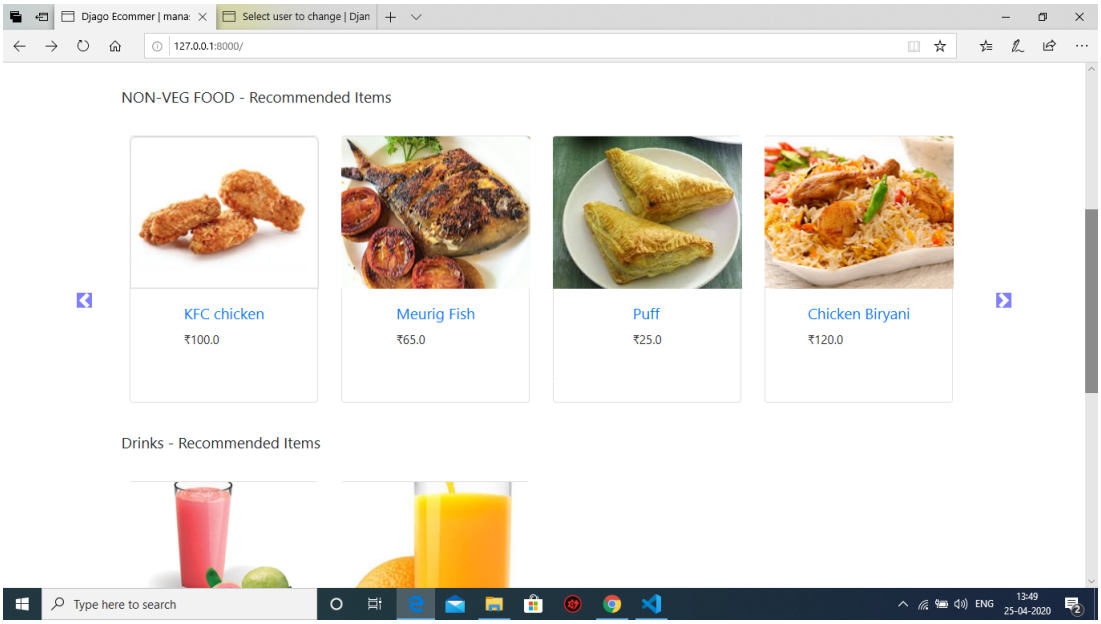


Figure B.3: Category Management Modue

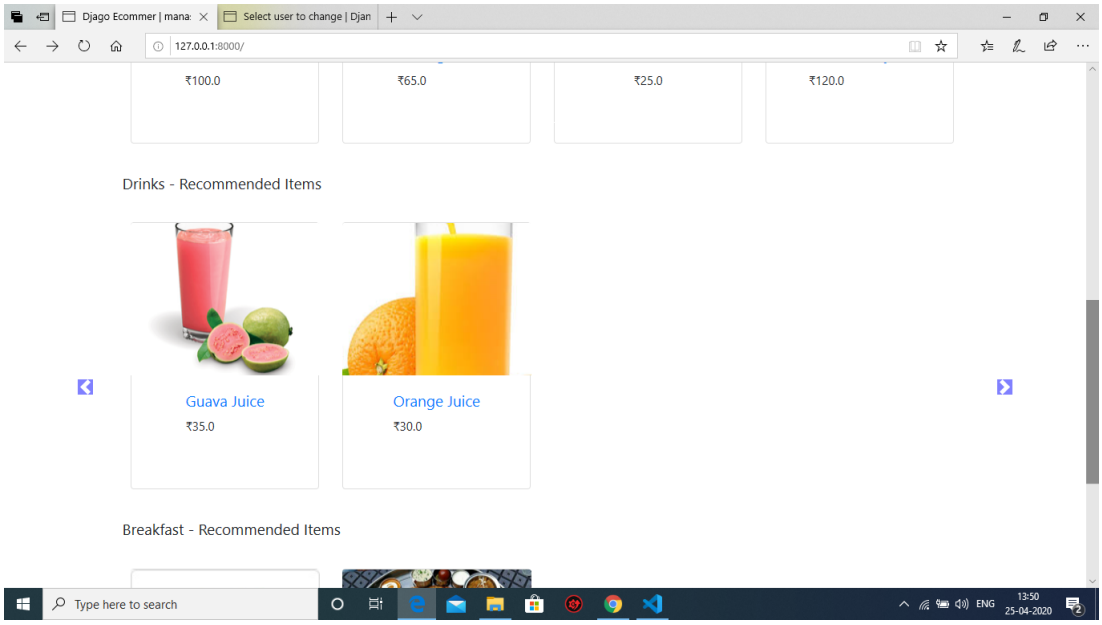


Figure B.4: Category Management Module

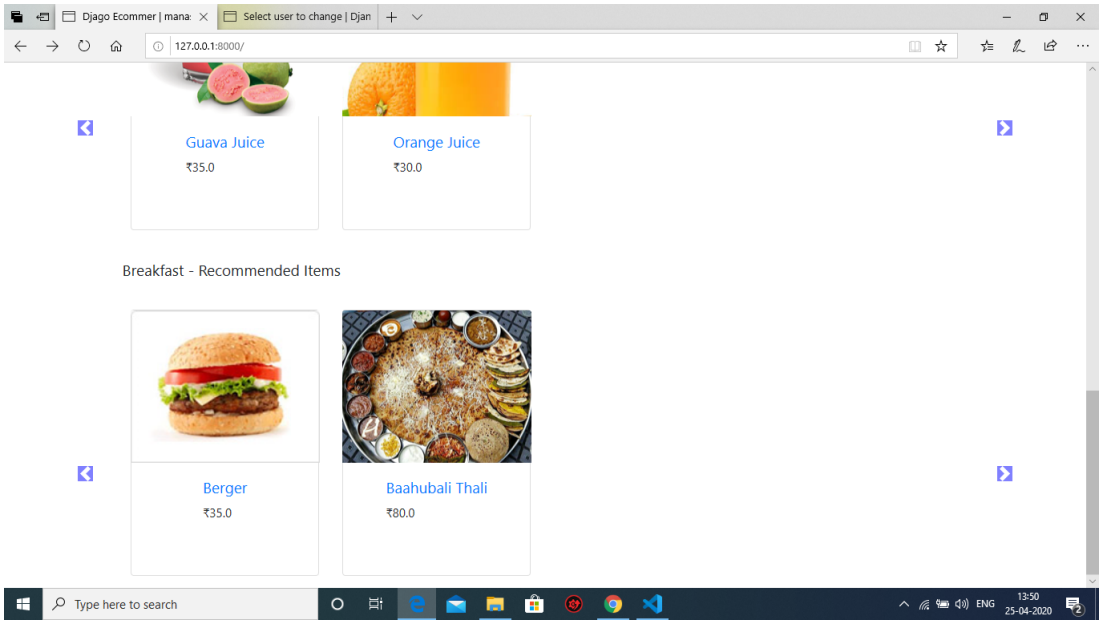


Figure B.5: Category Management Module

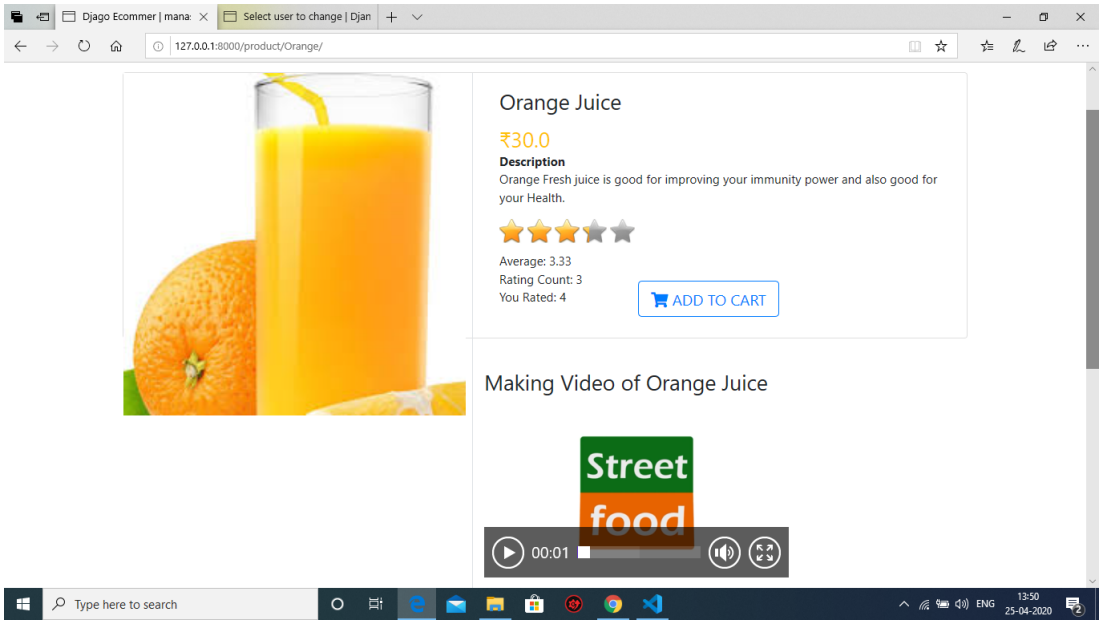


Figure B.6: Rating System

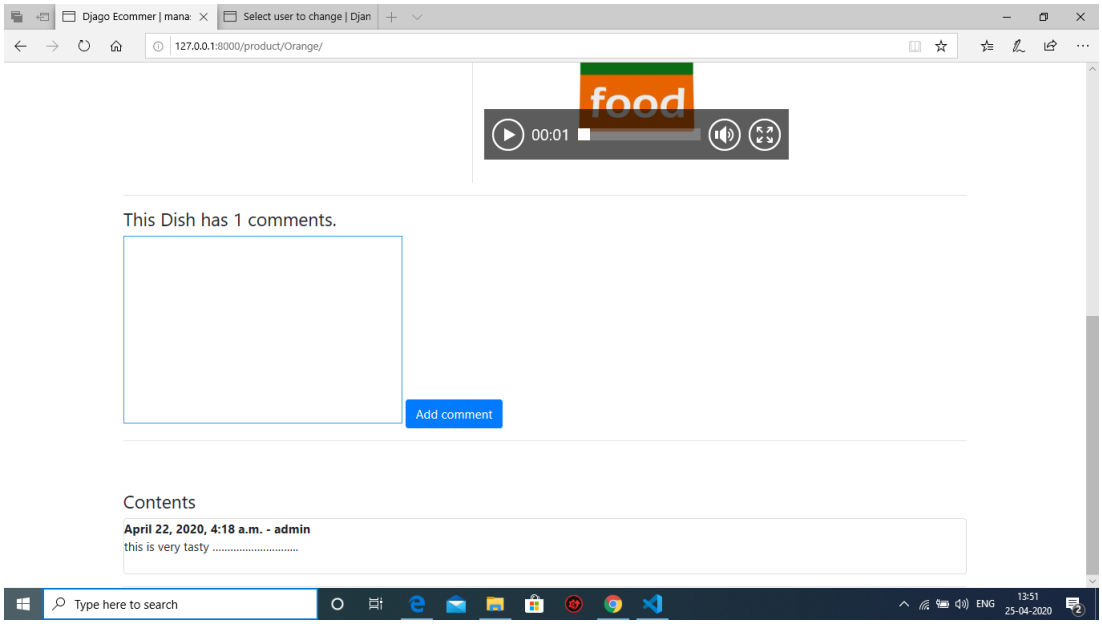


Figure B.7: Comment Area

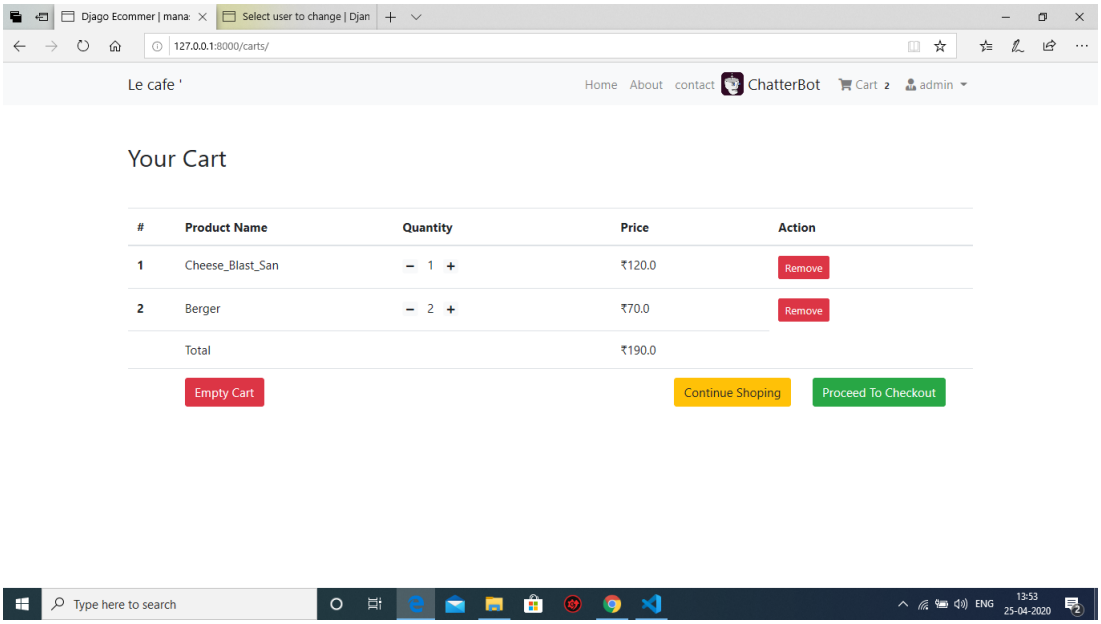


Figure B.8: Food Cart

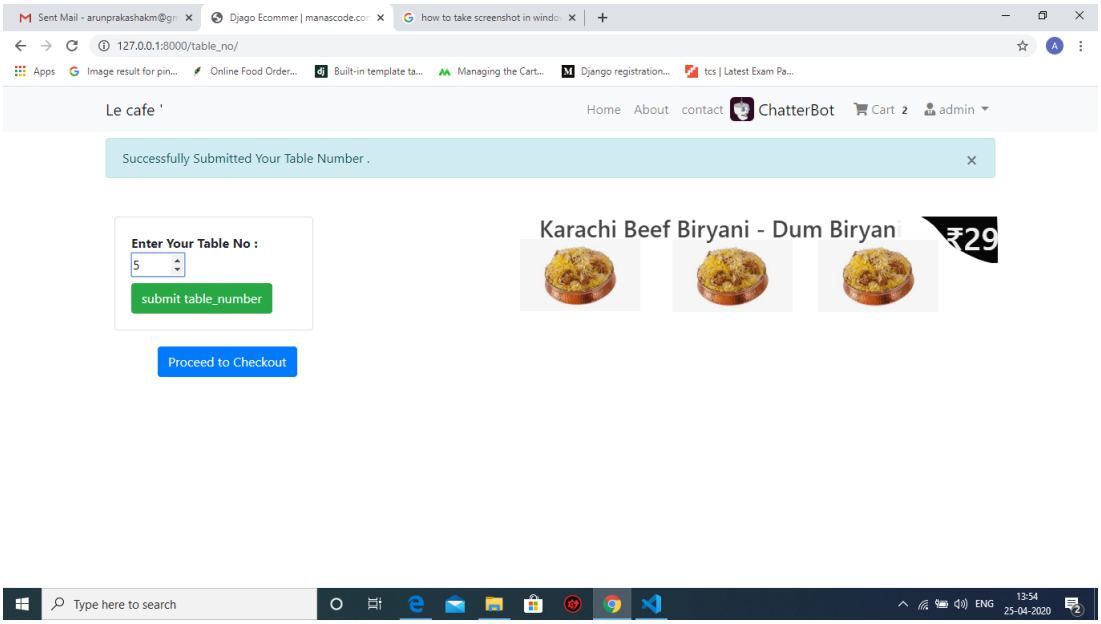


Figure B.9: Table Number Entry and Add-Ons

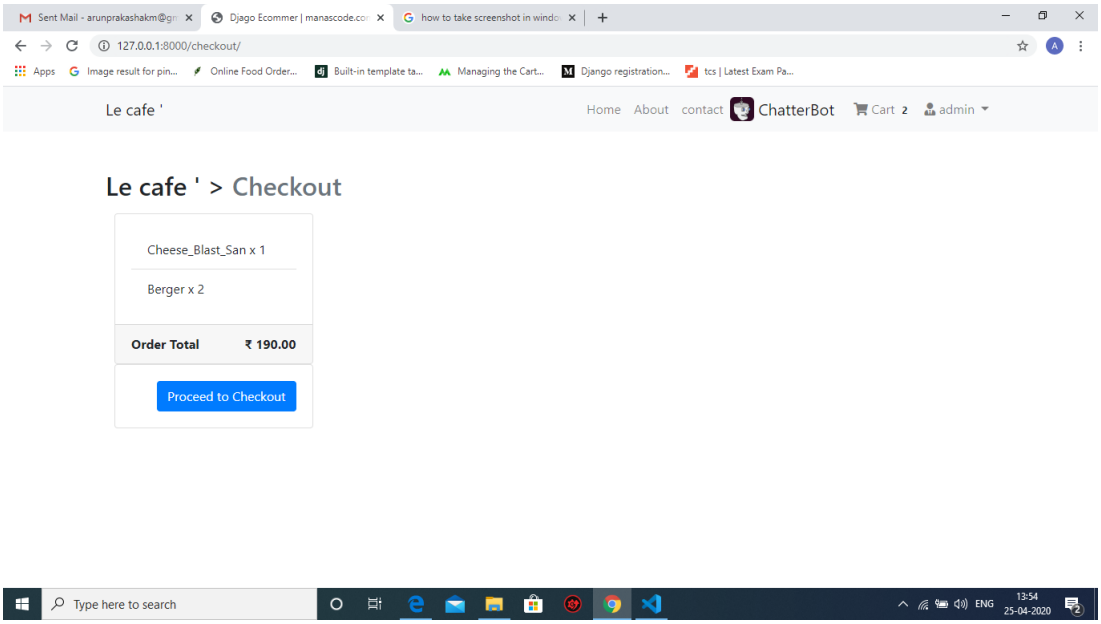


Figure B.10: Confirming Order

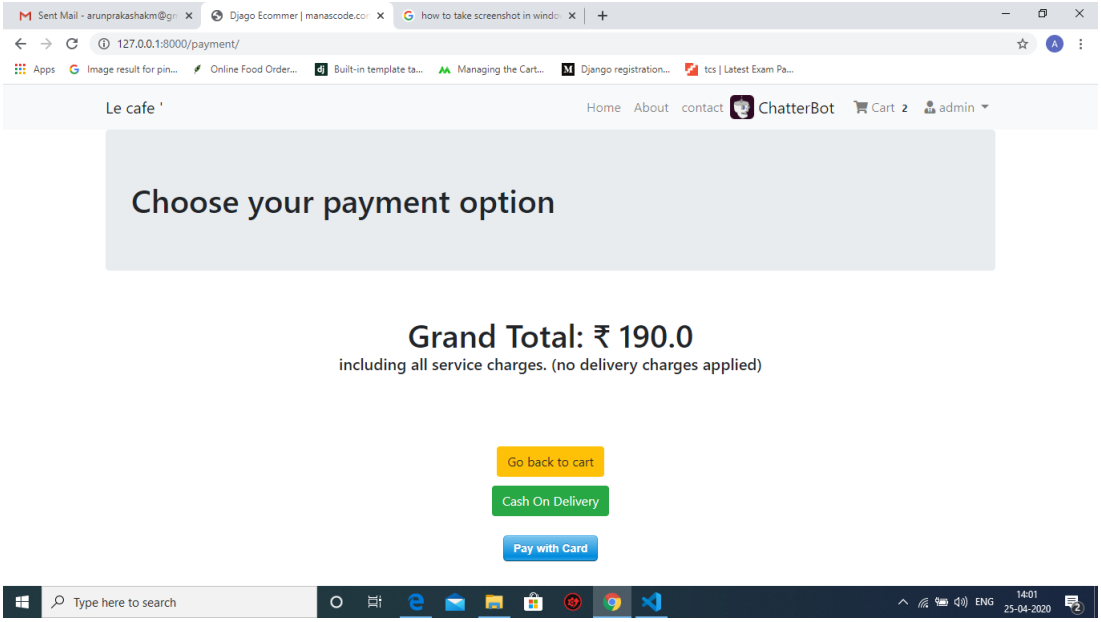


Figure B.11: Payment Module

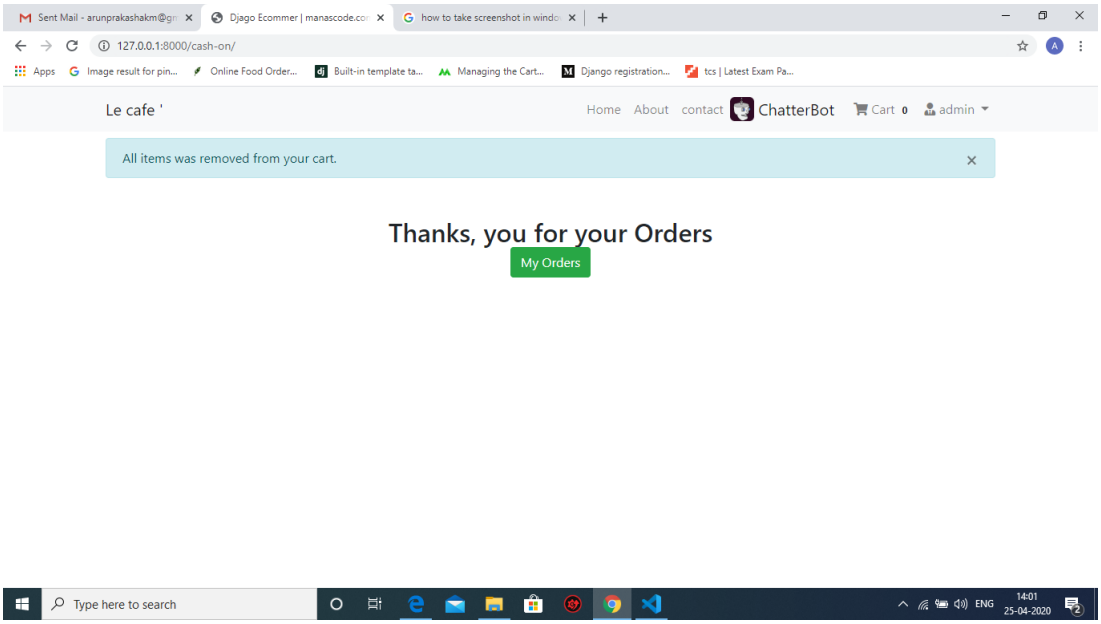


Figure B.12: Payment Module

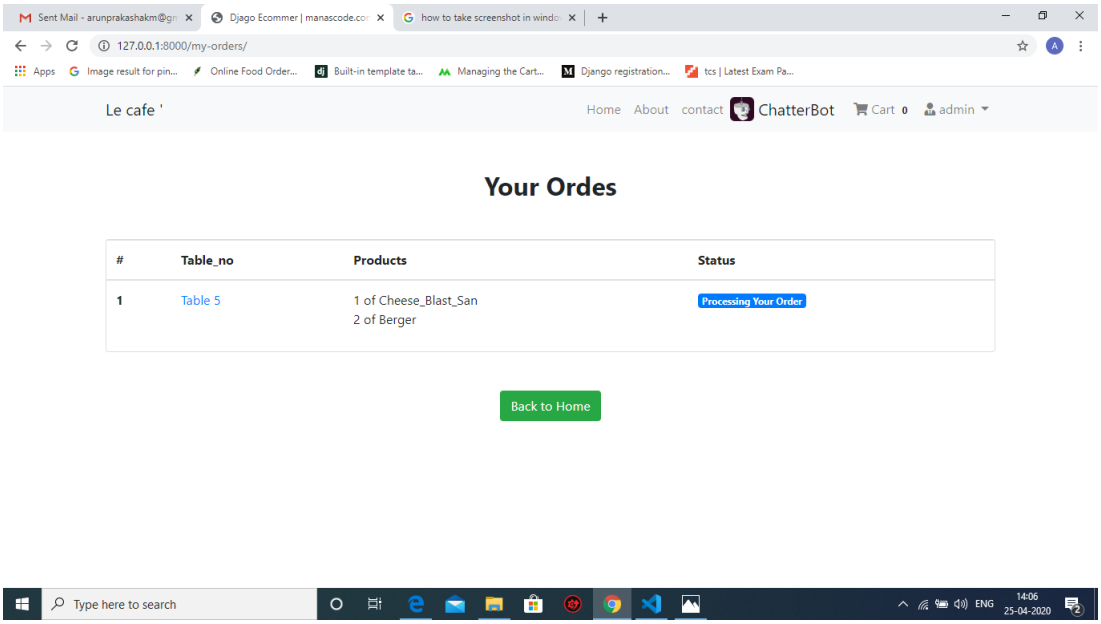


Figure B.13: Customer view of their placed Orders