National College of Ireland

Computing Evening Part Time Evening

Software Development

2022/2023

Kerrie Newman Ryan

X19135645

X19135645@student.ncril.ie

Communication Station

Technical Report

Contents

Executive Summary

The purpose of this report is to analyse and evaluate the implementation of the final year project "Communication Station". This project is aimed to develop a robust ordering and business management system. During development, this was upgraded from a singular software system that can only be used locally to a cloud-based online system. Using Python, JavaScript and the Django framework, the project was conceptualised with the increase demand of online ecommerce platforms. This platform was built with the idea of being utilised by small businesses and SMEs to massive corporations with buying, selling and managing all aspects of their business within this online marketplace.

The initial project idea was to create a singular service that could be used across all departments and assist with the running and managing of the business, this software could be used in any retail environment: gym, hotel, restaurant, or shop. During the development of the Communication Station platform, the overall management of the software would not be feasible, as it would have to be implemented on-site and cater for the specifications of the business itself, which would also make it difficult to test during the development. From a business aspect, this would not be profitable and would need many hours of testing and development to fine-tune this product. While keeping with the project brief a slight deviation was made in the outcome of the project development. An online ordering marketplace system was created. This report highlights the system, design, security and future outcome of this product. Leveraging Djangos framework built-in tools and Pythons flexibility this was implemented in structured phases. The first phase was learning and designing following the

development of the essential features of business registration, customer registration and management, product listing, cart management, order processing and payments.

Some of the major points for this include:

1. System Design: Model-View template architecture within Djangos framework allows for a scalable, secure and robust development.
2. Functionality: Business management tools, customer dashboard, service processing, payment management and cart management.
3. Security: The Django framework provides built-in Cross-Site Scripting protection, cross site request forgery, host header validation SQL injection projection, which are all required under OWASP guidelines for development. Authentication and authorisation processes incorporate to include password hashing and ensuring data security.
4. Integration: of payment gateways, map APIs and 3rd party services to for seamless application.
5. User Interface: easy to use application for both business and customers with an accessible interface.

In conclusion from the evaluation report, the final year computing project this system meets the core objectives of the computing project. This system fully aligns with industry standards while allowing for further scalability and deployment of the system.

Future recommendations for this would explore adding a customer support aspect for business or even integration an AI based support team to enhance user and customer experience. Also, monitoring, managing and updating the systems as needed for market demands and technical updates. Overall, the business online ordering system "Communication Station" is an efficient and reliable e-commerce system.

1.0 Introduction

1.1. Background

The change in marketing and online shopping, has required the digital revolution of businesses and updates in practices, and this is essential in today's competitive market. Many of the existing systems are expensive and complex, so the inception of this project was to allow better accessibility and convenience for both businesses and customers and also provide a cost-effective solution. Working with technical support in HubSpot, the motivation stems from the identified consumer needs for an accessible business management system with the availability of online ordering for small businesses and SMEs. The availability of a system like this for customers allows them to expand their services while opening up opportunities in new markets. The online system allows for the possibility of expansion nationally and internationally for the business, as well as providing job opportunities for managing the online system itself with developers and support representatives. The core idea behind this project is to provide an easy-to-use online system to help expand and manage the services of a business.

I wanted to use the opportunity to expand and use new technology and challenge myself with different programming languages, for this platform I used Python and Django frameworks. This allowed me to build my coding skills while allowing for a more reliable development cycle with reusable components that would align with the needs of a model business. This also allowed me the opportunity to create a real-world solution, the entire process from development to implementation gave me hands-on experience with the challenges of software development.

In summary, the evolving nature of e-commerce management systems allows this project further exploration in research and innovation, in terms of technological advancement and consumer data on behaviour and purchasing habits. This is why I undertook this platform development project.

Reference:

https://www.sciencedirect.com/science/article/pii/S0360835219306394?#bi005

1.2. Aims

The main aim of this project is to design and develop an online ordering system that allows for business management while simplifying the process of buying and selling through the internet via the online platform.

The main aims of this include:

- **Integration with existing systems:** develop a flexible architecture that allows integrations of inventory management and 3rd party payment processes.
- **User-Friendly Interface:** design intuitive and user friendly that enhances the ordering experience for the customer, allowing them to select, purchase and browse the products with ease. Thus permitting businesses to manage their outgoing products, inventory and orders
- **Secure Transactions:** Using built-in security measures from the Django frameworks that implement OWASP security measurements, ensuring the integrity of data and confidentiality for payment transactions.
- **Academic Endeavour:** Using real-world applications and techniques throughout development exploring new methodologies, techniques and tools that can be applied in future projects.
- **Scalability and Performance:** Creating a scalable system that can handle an increase number of businesses and customers ensuring consistent performance and reliability.
- **Compliance and Standards:** Ensuring to adhere to laws and regulations for regions related to e-commerce, privacy, region and security.
- **Accessibility for SMEs:** Providing an easy-to-use and affordable online order system that can be customised to suit the specific needs of businesses and customers.

- **Sustainability and Future Growth:** To build a platform that meets the current market demands but also for the improvement of the software system depending on demand and future technologies, keeping it competitive and up to date.

The aims of this project are beyond creating a function online business marketplace system, this also covers larger objectives, demonstrating a thorough strategy for handling the complex requirements of both enterprises and customers in the current e-commerce environment. The project aspires to significantly advance both the academic community and the larger industry by fulfilling these goals.

The objectives of this project go beyond merely developing an online ordering platform. They cover larger objectives for sustainability, usability, security, and accessibility, demonstrating a thorough strategy for handling the complex requirements of both enterprises and customers in the current e-commerce environment. The project aspires to significantly advance both the academic community and the larger industry by fulfilling these goals.

1.3 Technology

To create a user-friendly, efficient and high-performing online order and business management system called the "Communication Station", specific technologies were required during development to create and successfully operate platform for both businesses and customers. The technology used and required for this are list below:

1. **Programming Language: Python**

Python is a powerful, clear and concise language and has a wide range of libraries and frameworks available, allowing for secure authentication, database management, integrations and forms all of which have been used in the development o the "Communion Station". Pythons' robustness, scalability and ecosystem made it ideal for the development of this platform.

Development:

Server-Side Operations: All server-side functionalities will run on Python, facilitated by Django.

Data Processing: Managing CRUD operations for the system. For example:

Create new orders, add new items and categories,

Read order details,

Fetch menu land business listings,

View customer and business dashboards,

Update orders, menus and modify user profiles and dashboards,

Delete and cancel orders, and

Delete user information.

Data Validation: Python can confirm and validate the data of records or orders before saving it into the connected database. For example, checking the product availability before confirming the order.

API Integration: Python will handle the integration of Google Maps API, PayPal API, and PostgreSQL.

2. Framework: Django

Django is a high-level Python web framework with in built security features against common web vulnerabilities, like CSRF and XSS. Django is an ORM (Object Relational Mapping), allowing for data mapping into databases and allowing for data management and data integrity of the PostgreSQL database. This means it can also handle large datasets ideal for this development project.

Development:

Admin Interface: Django provides a customisable and back-end admin interface that can be customised and managed with the migrated data to the database. This allows orders, products, payments, users, and customer details can all be managed and maintained from the admin interface.

Authentication: To make the online buying system as safe as possible, an authentication system that is available within Django manages user registration, login, password reset, and more.

Back-End Capabilities: Django's middleware classes allow for processing requests and responses globally before they reach the view or after they leave the view. This can be useful for functions like request logging, user authentication, etc Django allows for the integration of APIs seamlessly and safely.

Database Handling: Design and manage database structure

3. System Database - PostgreSQL

PostgreSQL is an advanced, open-source relational database system.

Development:

Data Storage: PostgreSQL is better suited for both development and production since it supports larger datasets and has more sophisticated functions than SQLite. Using the built-in tool PgAdmin to assist with manging the database during development.

GDAL and PostGIs integration: this enables geospatial support in the database adding geographic objects to the related users in the database adding support for the geographic objects.

4. Communication - Gmail

Gmail is an email communication platform that works in according to Google Workspace

Development:

Notifications: Gmail will be used to send order confirmations, password resets, and other notifications to users.

5. Integration Google Maps API

Google Maps API Background: Using a JavaScript or Flash interface, developers can integrate Google Maps on websites using the Google Maps API.

Development:

Users can use location services to locate the closest outlet or store. Users can use the integration to auto-fill the address on their dashboard by just entering the postcode or location.

Delivery Tracking: Provide end customers with real-time access to delivery status.

6. Payment implementation - PayPal API

The PayPal API offers a way to include payment gateways in applications. This is a great secure option for adding a payment method on the marketplace communication station platform. It gives users more secure options for making a payment on an order

Development:

Payment Process: Handle all transactions ensuring a secure payment method for users.

Order Confirmation: Confirm orders after successful payments.

7. Geospatial handling - GDAL

A computer software library for reading and writing raster and vector geographic data formats is called the geographic Data Abstraction Library (GDAL).

Development:

Geospatial Analysis: GDAL will be used for the geospatial features, such as identifying delivery zones or forecasting delivery times based on geographic characteristics. Also, allowing for search functions based on the users and the business regions.

8. Front-End Development - HTML, CSS, JavaScript & Bootstrap

Web development relies on these core technologies. Web material is organised using HTML (HyperText Markup Language). This material is styled using CSS (Cascading Style Sheets). Interactive online elements are possible using JavaScript.

Development:

User Interface and Front End development: To make the online ordering system user-friendly and responsive, HTML and CSS will determine how it looks and feels. However, to improve interactivity, including form validation and dynamic content loading.

9. Project and version control - GitHub

GitHub is a cloud-based platform that uses Git for version control. This is used as version control to track changes in the project, allowing for collaborative work and ensuring the integrity of the codebase.

10. Miscellaneous

The Integrate Development Environment (IDE) used through development was Visual Studio Code (**VS Code)** due to its in-built libraries, easy-to-use extensions for Python imports and its in-built terminal. The in-build terminal allowed for development through a virtual Django environment, using the inbuilt Gitbash terminal, assisting the server-side function for testing the output of the communication station platform. Alongside this, VS code can handle different programming languages easily in the same environment and folder.

Font Awesome for the flexibility of styling icons on the platform, meaning specific buttons did not need to be hard-coded into the platform. .

The sweet alerts were integrated as well for a more stylish, responsive and accessible replacement to the JavaScript pop-up boxes for event handling and error notifications on the communication station platform.

Online ordering systems must be effective and user-friendly given the rise of e-commerce. This platform showcases an online ordering system created with Python's Django high-level web framework. Users may easily use the system's user interface while exploring products, managing their shopping carts, and conducting safe online transactions. The demand for effective, secure, and scalable online ordering platforms is rising as a result of the growth of digital commerce. This project combines Python's flexibility with Django's powerful features to produce a web-based ordering system that responds to market demands.

The combination of Django, which has Python at its core, PostgreSQL for reliable data processing, and APIs like Google Maps and PayPal guarantees a thorough and up-to-date technological approach. Each technology selected has a specific function, from user interaction to backend processing, ensuring the online ordering system is both effective and scalable.

1.4.Structure

In this section the overview structure of the document will be discussed.

Section One:

This section will go through the Executive Summary, Background, Aims and Technology. Each subsection will address work experience and where the initial idea, why I decided to undertake this project, the aims of the project and what I was looking to achieve during the development of the communication station platform. The list of technologies that were used during the development will also be discussed in this section.

Section Two:

Section two delivers an in-depth account of the technical development and system requirements that were used during the development of this project. This is the core section of the technical report, giving diagrams, use cases, architecture and design of the platform. Examples of the Graphical User Interface and the overall implementation of the platform with the testing process and evaluation.

Section Three:

Will address the conclusions of the Communication Station platform development and further research and future development for the platform. This will also include the references for the project and appendices.

2.0 System

2.1. Requirements

These measurable objectives give us a clear framework to make sure our online ordering system is deployed and used successfully by both customers and businesses.

User Registration and Authentication:

1. Any user that registers their details to join the online system can complete the process in under 3 minutes. Once the "Register" button is clicked, the page is redirected to where they must enter their details each form field is labelled clearly and will on accept the required information. For example, email address for the email field, phone number for the phone number field etc.
2. Over 99% of the users that register their details will be able to login in without error on their first attempt at registration.
3. Following the steps the user is labelled as a customer and only has access to purchase items on the online service and the customer dashboard.

Business Registration:

1. Businesses that are looking to sell and manage their business online using the communication station platform can complete the process in 3 minutes once they have their tax documentation available for the registration process.
2. Similar to the user registration process, each required form field is labelled clearly and only accepts the related information.
3. Once the information has been saved they will receive an activation link and over 90% of users can login on the first attempt.

Setting Up the User (Customer) Dashboard:

1. Once the user has logged into the customer dashboard they are shown a default image and cover photo where they have the option to update the images. In order, to achieve this they have to click on the "Account Settings" option on the left-hand side menu.

2. They can add their details and address where the Google Maps API will autofill their data and auto-fill their latitude and longitude to assist with searching businesses and products in the future.

3. The option for the update is not notified directly to the user, over 90% of users have been able to update their images and account information from the "Account Settings" on their first attempt without assistance. However, it has been noted that a knowledge base article to assist with new user set up speeds the process of customer profile creation and also assists with the reputation of the platform as a whole.

Setting up Business Profile from the Users Side:

1. The business profile takes more time to arrange and set up.

2. A short video of 10 minutes can assist businesses with setting up their account, we can share guides as well to assist with the set up of their account.

3. Over 80% of businesses can create and maintain their business accounts and dashboard on the first attempt without assistance however in some cases direct training and assistance from the communication station team is required. When this occurs we have a support team available to work with the business to assist with the set-up and development of their online profile.

Product/Service Browsing(Customers)

1. The potential customer can click the clearly labelled "Marketplace" button on the navbar and will be given the option to view the available businesses and services for that customer based on their region. From the saved address on their dashboard and the saved browser cookies that have saved the customers' location on their web browser.

2. When the potential customer after browsing the platform wants to make a purchase, they can click on the cart option and the system will display the options within seconds of the click.

3. Once the customer decides to add an item to their cart, they are then redirected to login to their account so that they can continue with the purchase.

4. Search results shall be displayed in under 3 seconds from the initiation of a search. The search bar appears on the main home page and is labelled in each field so that the correct information can be entered by the customer.

Order Process

1. The cart and payment options will only appear for logged-in customers. This occurs for over 99% of users and they do not need training or assistance to go product browsing on the platform.

2. The system will allow the customer to add an item to their cart from the marketplace item and will appear instantly in the cart.

3. Customers/Users will be able to view and modify their cart contacts with a delay of no more than 2 seconds.

Email Notifications:

1. All email notifications will be sent to users after taking the initial action. For example, once registered the activation email will be sent to the user instantly to allow for verification.

2. This is the same with order placement and password reset emails; all notifications of changes or updates are sent instantly.

Payments:

1. All payment transactions are handled through PayPal integration and are successful from the first attempt

2. Users will receive confirmation within 5 seconds from the pop-up window that the transaction has been completed.

2.1.1. Functional Requirements

1. User Management:

1.1 – Registration:

The system will allow new users to create an account using a valid email address and password.

Email addresses are unique for every user, so only one email address per user account.

Passwords are stored securely with hashing implemented on the user information for additional security.

1.2 Login/Logout:

Users can login and logout using their emails address and passwords.

Users can reset their password if needed for those that forgot their credentials.

Users can logout from the system, via their dashboard or navbar.

1.3 Business Management:

The system will allow businesses to create a business account using a unique email address, password and tax documentation

The business can create a business profile with an image and location of the business offering services.

1.4 Product Listing:

The system will allow businesses to add categories and lists of available products, with the relevant details for e-commerce sales e.g. name, image, price and description…

Clicking on the product will show all of the details and give the user the options to add to the cart.

The products can be filtered by categories and can be searched via the menu item and location concerning the user/shopper.

1.4 Shopping Cart:

Add to Cart:

Users should be able to add products to their cart.

The cart will update in real-time should the number of items and cost of items in the cart.

View Cart:

Users can view the items in the cart.

Users can modify their cart, they can increase or remove items as needed which will update in real-time as the interaction occurs.

1.5 Ordering:

Checkout Process:

Users can proceed from the cart page to the checkout

Users can add their billing and shipping information at the checkout

Users can pay using a verified and safe payment method (PayPal)

Ordering Confirmation:

Once the order has been successfully placed users will receive a confirmation message and email.

Order History:

Users can view their recent orders and past order history along with the status of that order.

Businesses can see their Order history and the amount of revenue they are making each month.

1.6 Payment System:

Integration of PayPal:

The payment system will accept the payment method safely and securely and not save the details of that payment on the browser or the customer dashboard.

1.7 Admin Management

Business Admin:

The business admin can edit, add and delete products services and categories on their account.

The business admin can view, edit and delete orders.

User Management:

The back-end admin can view and manage user accounts.

1.8 Responsive Design:

The Communication Station system should be accessible and usable on various devices, including desktops, tablets, and smartphones.

Non-Functional Requirements

1.9 Performance

The system shall support 10,000 concurrent users without exceeding a 5-second response time for primary functions.

System latency shall not exceed 3 seconds under peak load.

2.0 Security

All user passwords shall be hashed using an encryption method compliant with current industry standards.

System vulnerability assessments shall be conducted quarterly, ensuring no critical vulnerabilities exist.

2.1 Reliability

The system shall achieve a monthly uptime of 99.9%.

Database backups shall be executed daily with a successful restoration rate of 99.5%.

2.2. Scalability

Any increase of users or products by 25% shall not degrade system performance by more than 10%.

2.3. User Experience (UX)

90% of first-time users shall be able to place an order within 10 minutes of accessing the platform without assistance.

User interface loading times across devices shall not exceed 3 seconds.

2.4 External Interface Requirements

Database (PostgreSQL)

System functions making database calls shall have an average response time of under 2 seconds.

Data integrity checks shall be performed weekly with an error rate of less than 0.01%.

Payment Gateway (PayPal)

System integration tests with PayPal API shall be performed monthly, with a success rate of 99%.

95% of users shall complete transactions without manual assistance or re-attempts.

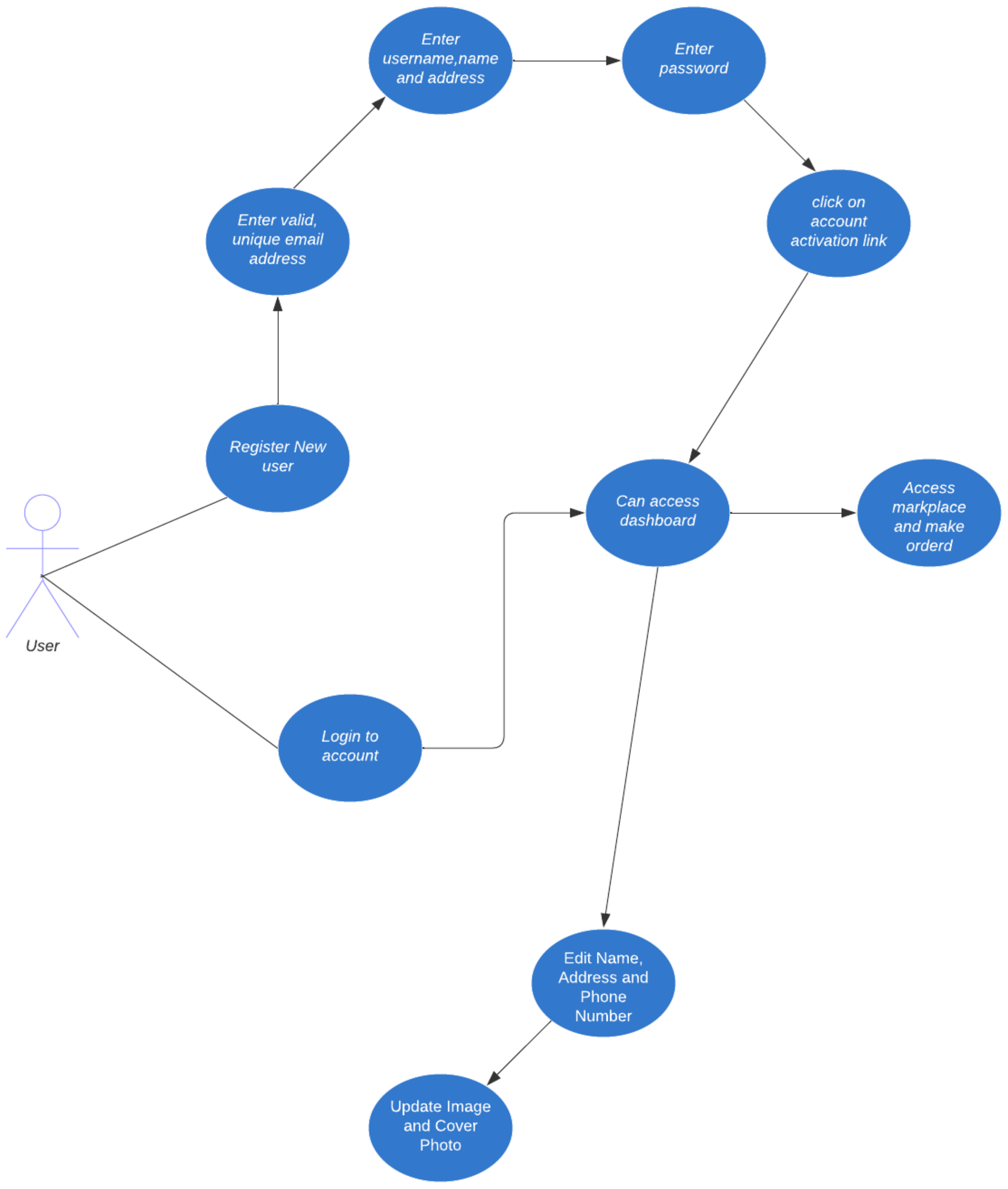
Email Service (Gmail)

Email delivery rates shall be maintained at 99.5% or above.

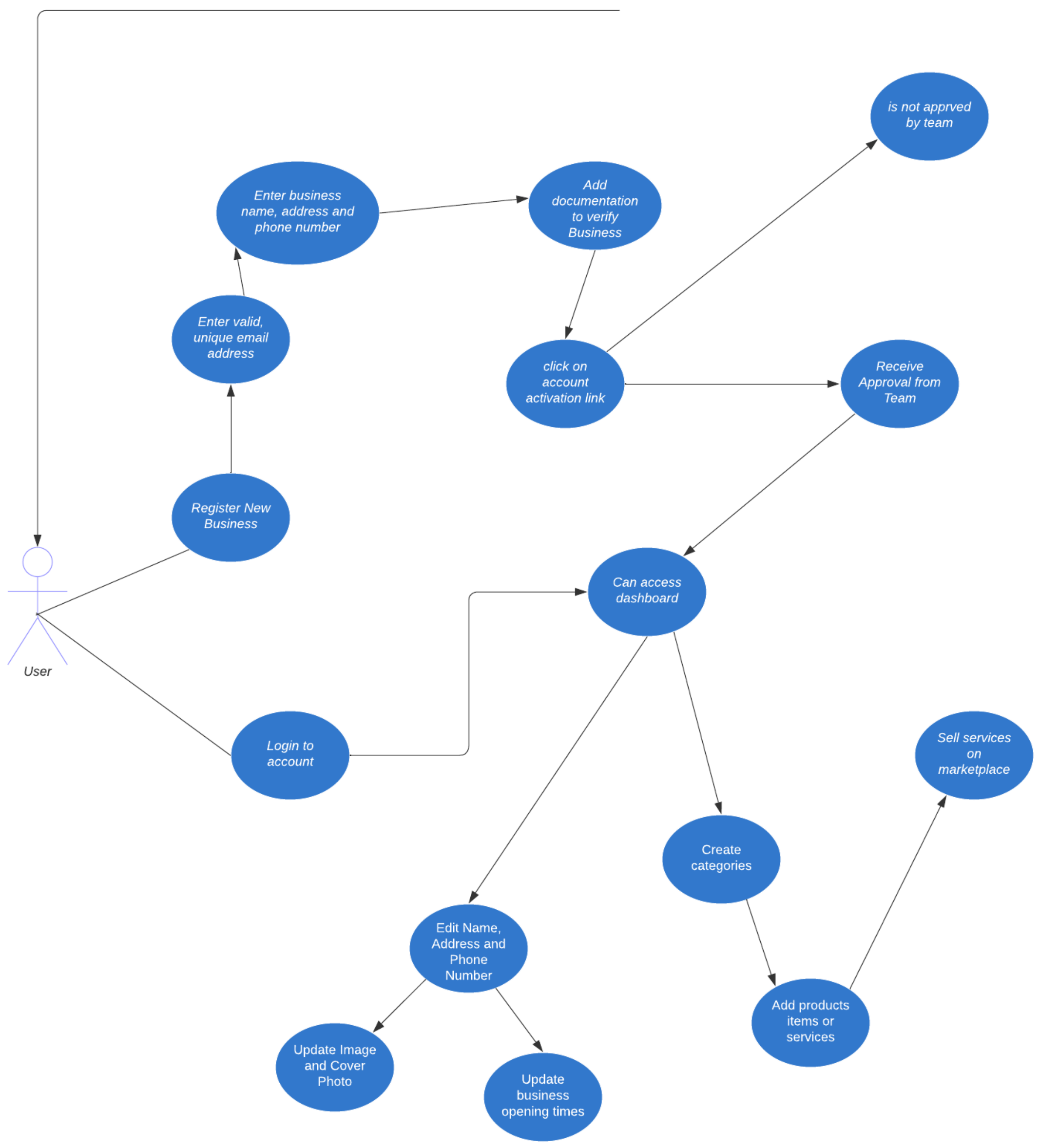
No more than 0.05% of emails sent shall be classified as spam by Gmail.

2.1.2. Use Case Diagram

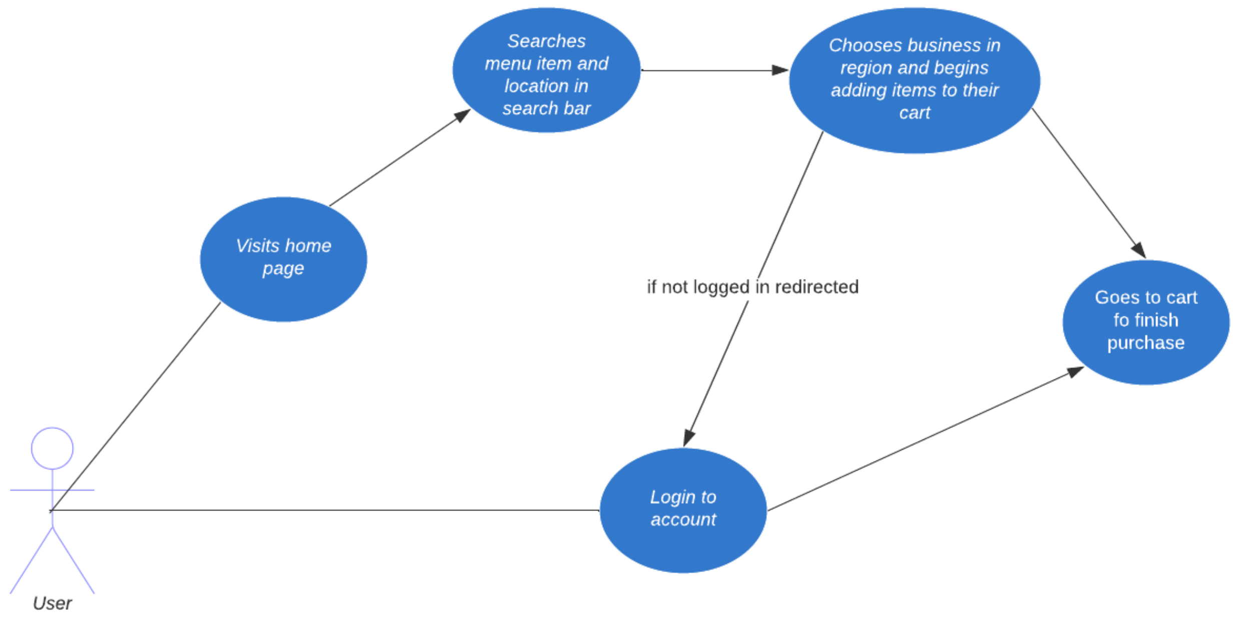
2.1.3. Requirement 1 – Login or Register New User


Enter username,name and address
Enter password
Enter valid, unique email address
click on account activation link
Register New user
Can access dashboard
Access markplace and make orderd
User
Login to account
Edit Name, Address and Phone Number
Update Image and Cover Photo

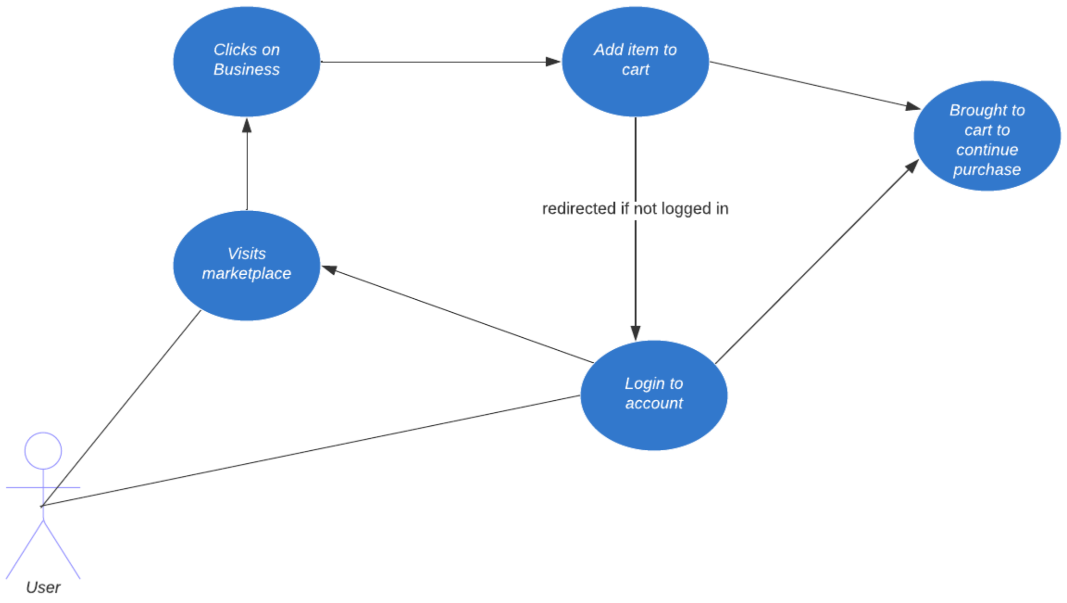
2.1.4. Requirement 2 – Login or Register New Business


User
Register New Business
Enter valid, unique email address
Enter business name, address and phone number
Add documentation to verify Business
click on account activation link
is not apprved by team
Receive Approval from Team
Can access dashboard
Login to account
Edit Name, Address and Phone Number
Update Image and Cover Photo
Update business opening times
Create categories
Add products items or services
Sell services on marketplace

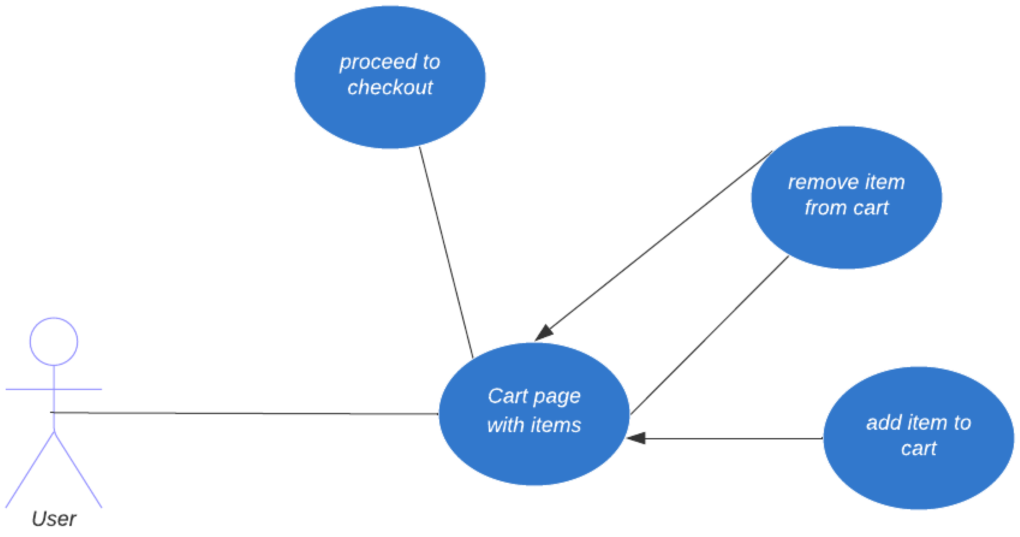
2.1.5. Requirement 3 – User Can search based on Geolocation


Searches menu item and location in search bar
Chooses business in region and begins adding items to their cart
Visits home page
if not logged in redirected
Goes to cart fo finish purchase
Login to account
User

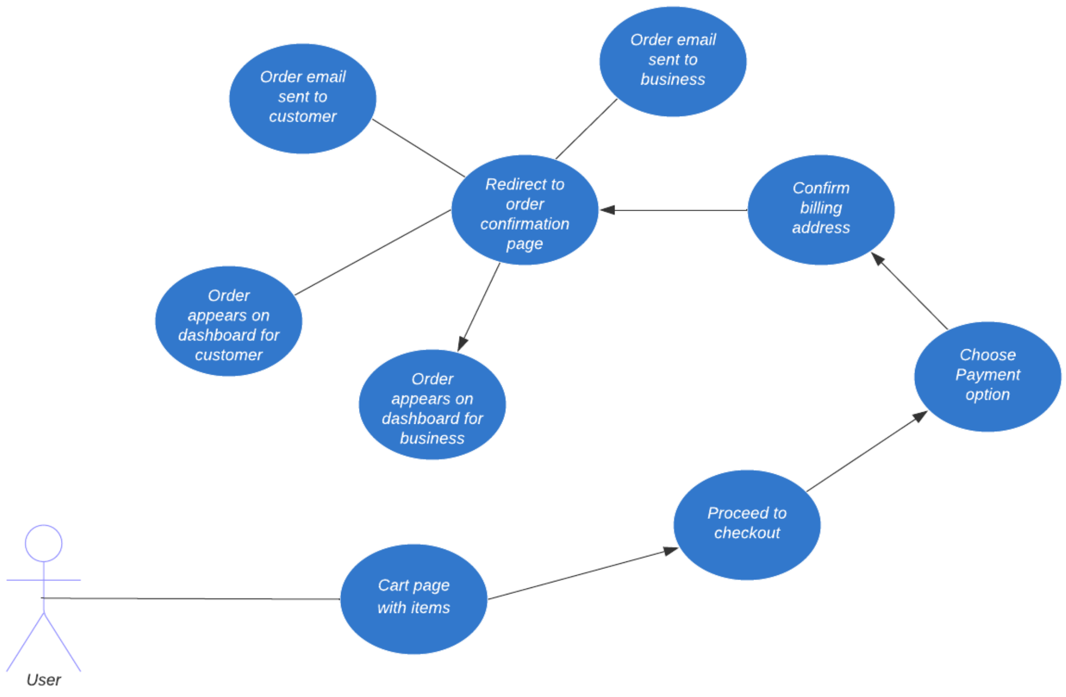
2.1.6. Requirement 1 – Visit Marketplace add item to cart



2.1.7. Requirement 4 – User can remove or add items to the cart



2.1.8. Requirement 1 – Confirm Order


Order email sent to customer
Order email sent to business
Redirect to order confirmation page
Confirm billing address
Order appears on dashboard for customer
Order appears on dashboard for business
Choose Payment option
Proceed to checkout
Cart page with items
User

2.1.9. Description & Priority

The priority levels ranges from High (H), Medium (M), to Low (L).

1**. User Registration and Authentication**

Description: The system will register new users and existing users can log in using a unique their email address and password. This requirement ensures the security of user-specific data, like cart items and order history.

This is high priority.

2**. Product Browsing**

Description: Users have the ability to browse through the list of available items. Each item will display details like name, image, price, and a description of the item.

Priority: Essential (E) - Core functionality that enables the primary purpose of the system.

3. Cart Function

Description: Users should have the ability to add or remove products to/from the cart. The cart should reflect the total price of added products.

This is high priority.

4**. Search Functionality**

Description: Users should be able to search for products using keywords or product names. This will help users find desired products quickly without browsing through the entire catalogue.

This is high priority.

2.2.Use Case

GENERAL CHARACTERISTICS	
Use Case #1	
Scope	User case is to outline the business registration process for the "communication Station" business and ecommerce platform. Following the approval and authentication of the business.
Level	User Goal
Description	This use case describes the journey of a business that is interested in using the online business platform. Registering their business. Providing tax documentation to authenticate the business and awaiting approval from the platform administrator
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status	Completed
User Cases:	Register Business, Upload tax certificate, Admin approves the business and the business profile is created
Primary Actor	Business Owner
Secondary Actors	Marketplace Administrator
	Flow Description
Preconditions	- The actor has a valid business with tax documentation and is looking to sell products online.
Assumptions	- The actor is a valid business with legal documentation and has access to the registration page for the platform.
Trigger	- The actor opens the home page and clicks on the red button labelled "Register Business" .
Success Post Condition	- Businesses details are saved and the business account is saved - After the information has been saved an alert pops up on the bottom corner stating "Your business has been saved successfully and is now under approval". - The actor is redirected back to the registration page. - Once the business has been approved they are sent an approval email. - The actor can then login and begin creating their business profile.
Failed Post Condition	- The email address has been used and will be sent an error message "User already exists please log in to your account". - The actor is given an error message that "passwords do not match" if the passwords do not match and the actor must enter their information again.
Normal Flow:	1. The business owner navigates to the registration page on the e-commerce platform. 2. The business owner fills out business details (e.g., name, address, contact info, name of the business).

	3. Business owner uploads tax certification. 4. Business owner clicks "Submit" for registration. 5. The system sends the registration for admin approval. 6. Business owner receives an email acknowledgement with a message about the pending approval. 7. Business owner is redirected back to the registration page.
Alternative Flow:	A1: Mandatory Field Is Missing The Business Owner is requested to fill up any blank fields that are required before submitting if any are missing. The owner of the business adds any blanks. At the second location in the main flow, the use case continues.
Exceptional Flow:	**Email Address Already Used in Exceptional Flow E1** Previous use of the email address is discovered by the e-commerce system. Error message: "User already exists; please log in to your account." is given to the business owner. **E2: Incorrect Password** The e-commerce system recognises that the passwords submitted do not match. "Passwords do not match," is the error message that appears and prompts the business owner to re-enter the information.
Termination	The software displays logical alternatives, such as browsing products, configuring profile information, or managing business listings.
Post Situation	While awaiting additional user interactions or procedures, the system enters a wait state.
Overview	This use case outlines the process a business must follow to register their business on the "Communication Station" platform to mange their business online. During the registration process, the actor must enter their detailed information and provide a valid tax certificate to show proof of business. After the successful submission, the platform admin has to investigate and authenticate the business information. Once authenticated they can provide approval to the business to begin operating on the platform. This use case ensures that only legitimate businesses are allowed to sell products and services on the communication station platform. While

	also going through the process that the business must complete to register the business on the account.
GENERAL CHARACTERISTICS	
Use Case #2	
Scope	To register a new customer to buy and purchase on the communication station system with a password confirmation and follow-up email to allow for account activation
Level	User Goal
Description	This user case describes the customer creating account, registering the account and receiving the confirmation email allowing for the activation of the account.
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status	Completed
User Cases:	Customer, System and Gmail communication email system
Primary Actor	Customer/User
Secondary Actors	n/a
	Flow Description
Preconditions	The customer has a valid email address and wants to create a new account. The home page is initialised and awaiting ready to accept a new user
Assumptions	This user indicates a desire to register for an account on the online store.
Trigger	The user opens the home page and clicks on the "Register" button
Success Post Condition	1. A registration form with fields for a name, email address, and password is displayed by the system. 2. The User completes and submits the registration information. (If the form is filled out improperly, refer to A1 for an alternative flow.) 3. The data is verified by the system, and a pending account is created. The Email System simultaneously sends a confirmation email that includes an activation link. (If email cannot be sent, see E1.) 4. The user clicks the email's activation link. Flow Alternate A1: Wrong Form Submittal

	- 1. Fields that are incomplete or lack mandatory information are flagged by the system. - 2. The User updates the form and submits it again. - 3. The use case continues at the third main flow position.
Failed Post Condition	The email address has been used and will be sent an error message "user already exists please login to your account". The actor is given error message that "passwords do not match" if the passwords do not match and the actor must enter their information again.
Normal Flow:	1.The actor clicks on the "Register" on the top right hand corner of the home page. 2.The system presents a registration form with fields for name, email, and password. 3.The user fills in the registration details and submits the form. (See A1 for alternate flow if the form is filled incorrectly.) 4.The system validates the information and creates a pending account. Simultaneously, the Email System sends a confirmation email with an activation link. (See E1 if email fails to send.) 5.The User clicks the activation link in the email.
Alternative Flow:	**Alternate Flow A1: Incorrect Form Submission** 1.The system identifies fields that are filled out incorrectly or missing mandatory information. 2.The User corrects and resubmits the form. 3.The use case continues at position 3 of the main flow.
Exceptional Flow:	**Exceptional Flow E1: Email Sending Failure** The system detects the failure in sending the email. The User is informed about the failure and is given the option to resend the confirmation email or change the email address.

	The use case continues at position 4 of the main flow if the User chooses to resend the confirmation email.
Termination	The system presents the next logical options, such as logging into the account, browsing products, or setting up profile details.
Post Situation	The system goes into a wait state, waiting for further user interactions or processes.
Overview	This use case outline the process of a customer/user would have to follow in order to register their customer account on the "Communication Station" platform to order goods and services. During the registration process the actor must enter a valid email address and phone number. After the information has been added they will receive an email activation link and brough to the login page for their account.

GENERAL CHARACTERISTICS	
Use Case #3	
Scope	This use case outlines the procedure for a user searching for items based on location within the "Local Finds" ecommerce platform.
Level	User Goal
Description	This use case describes the flow for customers and users to search for businesses in order to purchase online and filtering search results according to geographical proximity.
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status	Completed
Use Cases:	Open Home Page, Enter Search Query, Filter by Location, View Local Listings
Primary Actor	User and potential customer
Secondary Actors	System and GDAL integration
	Flow Description
Preconditions	The user has access to the platform and has intent to purchase items.
Assumptions	The actor has access to the marketplace platform and the system is able to access the location of the user from their web browser. The system has listings with geographical information. The user wishes to shop locally.
Trigger	The actor accesses the home page and begins a search query on the main page.
Success Post Condition:	User is presented with a list of items available locally, based on their set location or detected location. A message is displayed: "Showing x number of businesses”
Failed Post Condition:	No local listings found, and a message is displayed: "No local items found for your query. Try expanding your search radius."

Normal Flow:	1.Actor navigates to the home page of the ecommerce platform. 2.Actor enters the item that they wish to purchase, the location of the business an the radius of 5kn, 10km or 15 km within the actors location. 3. System displays search results, prioritising listings near the user's location. 4.System shows how far the businesses are from the actors current location 5. Actor browses through local listings.
Alternative Flow:	**A1: Modify Search Radius** The actor adjusts the radius for local search to expand or narrow down the search results based on distance. System updates the business listings accordingly.
Exceptional Flow:	**E2: No Local Listings Found** The system doesn't find any listings within the user's specified or default radius. Message displayed: "No local items found for your query. Try expanding your search radius."
Termination	After browsing listings, the user can choose to view a product in detail, continue searching, or navigate to other sections of the platform.
Post Situation	System awaits further user interactions or queries.
Overview	This use case details the process for users wanting to discover and support local businesses or sellers by shopping for items based on proximity. The "Local Finds" platform emphasizes community commerce by letting users prioritize geographically close listings. Filtering items by location ensures that users can readily find and purchase products in their locality, fostering local commerce and reducing shipping-related environmental impacts.

GENERAL CHARACTERISTICS	
Use Case #4	
Scope	This use case outlines the case to add an item the actor wishes to purchase to their cart.
Level	User Goal
Description	The use case details the actors' journey of adding items to their shopping cart. This includes different outcomes depending on their login status.
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status	Completed
Use Cases:	The actor has chosen a business that they wish to purchase a product from. The have navigated to the Product page, Selected the item and Added it to their Cart
Primary Actor	Actor Customer/Shopper
Secondary Actors	Online Marketplace System, Database
	Flow Description
Preconditions	The user has access to the platform and wants to purchase items from a business that is active on the platform
Assumptions	The system has an active cart system and the user//actor can see in real time the cart increase and decrease when adding and removing the items from the cart
Trigger	The actor finds an item online that they wish to purchase and wants t add it to their cart
Success Post Condition:	The Item is successfully added to the cart. The number of products purchased shows on the product increases from 0 to 1,2 etc

	The shopping cart icon on the top right hand corner that displays a "0" increases by one each time the item is added to the cart.
Failed Post Condition:	An error message will appearing showing the user that "The Item does not exist in their cart"
Normal Flow:	**(User is Logged In):** 1.Actor navigates to the desired business and chooses the product from the category and menu page. 2.Actor selects the desired product or service and the number of items that they are looking to request. 3.Actor clicks the "+" icon beside the product or service image. 4.System adds the item to the user's cart associated with their account. 5. The shopping cart increases the number by one in the on the UI, confirming that the item has bee added to the cart
Alternative Flow:	**A1: Add to Temporary Cart/User has not logged in or signed up** 1. Actor navigates to the desired product page. 2.Actor selects the desired product and the amount that they want to order 3.Actor clicks "+" button. 4.System adds the item to a temporary cart (session-based). 5. Actor is redirected to the login page, where the actor is advised to login to their account via the pop up message "Please login to continue"
Exceptional Flow:	**E1: System Error** A technical glitch prevents the item from being added to the cart. Message displayed: "The item is not in your cart please try again." **E2: Session Timeout (For logged out users)** The user has been idle for a long duration and their session has timed out. When they try to add an item to the cart, their temporary cart has

	been cleared. Message displayed: "Your session has expired. Please refresh the page and try adding the item again."
Termination	After adding the item, the user can choose to continue shopping, view their cart, or proceed to checkout.
Post Situation	System awaits further actor interactions or actions on the platform
Overview:	This use case describes a required marketplace interaction of adding items to the shopping cart. The platform must ensure this action is seamless for both logged-in users, ensuring persistent storage of their cart, and logged-out users, for whom a temporary cart is used. The system will only show the available stock and services for the businesses available, where when the category or product is not available its removed from the menu via the businesses management dashboard.

GENERAL CHARACTERISTICS	
Use Case #5	
Scope	This use case outlines the scenario where the actor removes an item from their cart that has already been added and they are no longer interested in purchasing
Level	User Goal
Description	The use case shows the actors journey of removing the unwanted or unneeded items from their shopping cart. This is all dependent on the actors login status
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status:	Completed
Use Cases:	The actor has chosen a business from which they previously intended to purchase a product. They navigated to the Product page, Selected the item, Added it to their Cart, and now they want to remove it.
Primary Actor	Actor/Customer/Shopper
Secondary Actors	Database and System
	Flow Description:
Preconditions:	The actor has access to the platform and has previously added items to their cart from an active business on the platform.
Assumptions:	The system has an active cart system. The actor can see in real-time the cart increase and decrease when adding and removing items.
Trigger:	The actor decides they no longer wish to purchase a specific item and wants to remove it from their cart.
Success Post Condition:	The item is successfully removed from the cart. The number of products in the cart decreases accordingly. The shopping cart icon on the top right-hand corner that displays the number decreases by one each time an item is removed from the cart.
Failed Post Condition:	An error message appears showing the actor that "Item is not in your cart"
Normal Flow:	**User is Logged in:** 1.Actor navigates to their cart. 2.Actor identifies the product they wish to remove. 3.Actor clicks the "-" icon beside the product to reduce the quantity of items in the cart or clicks on the red "bin" icon to completely delete the product from their cart.

	4.System retroactively removes the item from the actor's cart associated with their account. 5.The shopping cart number decreases by one in the UI, confirming that the item has been removed.
Alternative Flow:	**A1: The item is fully deleted from the checkout using the "red bin" delete icon** The actor clicks on the icon and the item an all of the added quantity I fully removed and deleted from the cart. The message "Successfully removed from your Cart" appears to confirm that the action was successful. **A2: The actor is not logged in to their account** The shopping cart icon does not appear on the navbar until the user logs in. Once logged in the number of items in the cart will appear for the actor
Exceptional Flow:	**E1: System Error** A technical glitch prevents the item from being removed from the cart. Message displayed: "This item does not exist in your cart . Please try again."
Termination	After removing the item, the actor can choose to continue shopping, view their modified cart, or proceed to checkout.
Post Situation	System awaits further actor interactions or actions on the platform.
Overview	This use case describes an essential marketplace interaction of removing items from the shopping cart. The platform must ensure this action is straightforward for both logged-in users, ensuring persistent storage of their cart modifications. The system automatically reflects the update on the checkout and cart user interface. If a category or product becomes unavailable, it's removed from the menu via the business management dashboard from the business end.

GENERAL CHARACTERISTICS	
Use Case #6	
Scope	This use case describes the functionality of updating the cart's total amount in real-time based on the addition or removal of items. The service and
Level	User Goal
Description	The use case shows total amount of the shopping cart updating dynamically as the items are been added and removed
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status:	Completed
Use Cases:	
Primary Actor	Actor/Customer/Shopper
Secondary Actors	Database and System
	Flow Description:
Preconditions:	The actor has access to the platform, is logged into their account and can add or remove products from their cart. Prices of the products are updated and accurate in the system.
Assumptions:	That the actor is logged in and is shopping or order using the platform
Trigger:	The actor adds or removes an item from the cart
Success Post Condition:	The cart total amount is updated and correctly reflects the sum of the prices of the items in the cart. The total tax and service fee updated in real time as the items are been updated If all items are removed, the message "Cart is empty" is displayed, and the total amount shows as €0.
Failed Post Condition:	The handling exception shows the message “Cart is empty. Please Try Again”.
Normal Flow:	**(User is Logged In):** 1.The actor adds or removes a product from their cart. 3.The system retrieves the product price from the database and updated the amount on the cart total amount 4.The system recalculates the cart's food tax and service fee and updated he carts total amount. 5.The updated amount is displayed in the cart's summary section.

	6.If all items are removed, the cart displays the message "Cart is empty" and the total amount shows as €0.
Alternative Flow:	**A1: Actor adjusts the quantity of a product in the cart** Actor changes the quantity of a product (either increasing or decreasing). The system recalculates the cart's total based on the new quantity and the unit price of the product. The new total is displayed in the cart summary.
Exceptional Flow:	**E1: System Error during cart total recalculation** Due to technical glitches or database retrieval issues, the cart total fails to update. Message displayed: "Item in cart does exist please try again. "
Termination	Once the cart's total is updated, or the cart is cleared, the actor can decide to proceed to checkout, continue shopping, or close the platform.
Post Situation	The system awaits further interactions or actions from the actor.
Overview	This use case provides a comprehensive outline of how the online marketplace system ensures that actors are always informed about the total amount of their purchases in the cart and that it is updated in real time for the actor. The total amount is a vital piece of information that affects purchasing decisions. It's crucial for the system to maintain accuracy and promptly display updates to enhance user trust and satisfaction

GENERAL CHARACTERISTICS	
Use Case #7	
Scope	This use case describes the functionality of a customer logging on and viewing the customer dashboard in the system. Giving them the option to update their profile and view their order history.
Level	User Goal
Description	The use case shows the process through which a user logs into their account and accesses the customer dashboard. Where the options are to update their current details on their account. Updated their profile picture, cover photo, address and phone number
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status:	Completed
Use Cases:	View dashboard, Update Profile and View Order History
Primary Actor	Registered Customer/User
Secondary Actors	Database and System
	Flow Description:
Preconditions:	The user has a registered and activated account and can log into the communication station platform. The user can log and access their account. All of their information is stored and saved on the database
Assumptions:	That the actor is logged in and wants to view their order history and profile information
Trigger:	The actor chooses to login to their account
Success Post Condition:	The user can views their recent order history. The user is able to updated their profile details
Failed Post Condition:	The handling exception will route the customer back to the main dashboard page
Normal Flow:	**(User is Logged In):** 1.The user clicks on the “Account Settings” button on the side bar of the dashboard. Or clicks the “Edit Profile” 3.The user sees the options to “Update the profile image”, “Update Cover Photo”. When clicked the user is prompted to add a new photo from their device. 4.The user can edit their needed fields 5.After the changes have been made the system validates the new information, updates the database and is give a success message to verify they information is now saved.

	6. the new information and images are now appearing on the customer dashboard. **View Recent Order History:** The dashboard shows the 5 most recent orders for the customer and the complete total of order that they have made in the past.
Alternative Flow:	**A1: Attempting to Update Without Selecting an Image** 1.Within the 'Update Profile' section, the user clicks on 'Change Profile Picture' or 'Change Cover Photo'. 2.The user is prompted to upload a new image or select from existing ones. 3.The user bypasses the selection and tries to save without adding an image. 4.The system detects that no image has been selected or uploaded. 5.An error message is displayed: "No image selected. Please upload Profile of Cover Image" 6.The user is prompted again to select or upload an image before saving the changes.
Exceptional Flow:	**E1: Incorrect Login Credentials** 1.User enters incorrect login details. 2.System displays "Invalid username or password. Please try again." message. 3.User can retry login, reset password, or create a new account.
Termination	The user can log out of their account or proceed o shopping on the communication station platform
Post Situation	The system awaits further updates
Overview	This revised use case provides a more detailed insight into how registered users can access and comprehensively update their profiles on the online ordering platform. Incorporating visual updates along with primary contact details ensures that users have the freedom and flexibility to personalize their online presence, enhancing user experience and engagement on the platform.

GENERAL CHARACTERISTICS	
Use Case #7	
Scope	This use case describes the functionality of the business status open or closed, appearing on both the marketplace via a banner showing open or closed. Also the opening hours of the related business on the page.
Level	User Goal
Description	The use case shows the available businesses
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status:	Completed
Use Cases:	Display the business hours on the business profile and inform user of business status
Primary Actor	Registered Customer/User /
Secondary Actors	Database, System, and Business Owner
	Flow Description:
Preconditions:	The business has specified its weekly hours of operation in the system. The system can detect the current day of the week and show the opening hours accordingly to that
Assumptions:	The available businesses appear on the marketplace
Trigger:	The user is browsing the marketplace and wants to know if a particular business is currently open.
Success Post Condition:	The user is accurately informed of the business's current operational status through a displayed banner.
Failed Post Condition:	The business does not show the opening times, the business does not appear closed or open.
Normal Flow:	**User is Browsing the Marketplace):** 1.The system detects the current day of the week. 2.The system retrieves the operating hours for businesses from the database and shows this for the user. **For the specific day:** 1.If it's Monday: The system determines that the business is closed. An overlay or banner with "Closed" is displayed on the business's section or page on the marketplace. 2.If it's Tuesday: The system determines that the business is open. An overlay or banner with "Open" is displayed on the business's section or page on the marketplace.

	This flow continues for other days of the week based on the business's set schedule. The user views the business's status and decides to engage accordingly.
Alternative Flow:	**A1: Business Changes Regular Operating Hours** 1.The business owner updates their hours of operation in the system (e.g., due to a holiday or special event). 2.The system captures and saves this change in the database. 3.When a user views the business, the updated hours and current status (Open or Closed) are displayed accurately on the page from the system.
Exceptional Flow:	**E1: System Error in Retrieving Business Hours** 1.Due to a technical glitch or database retrieval issue, the system cannot determine the business's status. 2.No hours will appear and the business will not appear open or closed. Once refreshed the opening hours will update.
Termination	After viewing the business status, the user can decide to explore the business further, continue browsing other businesses, or leave the marketplace or the business page.
Post Situation	The system awaits further interactions or actions from the user.
Overview	This use case offers a clear representation of how the online marketplace system efficiently and accurately communicates the operational status of businesses to users. By promptly and correctly displaying a business's open or closed status, the platform provides a better browsing experience and ensures that users can make informed decisions while shopping or seeking services.

GENERAL CHARACTERISTICS	
Use Case #7	
Scope	This use case describes the functionality of creating an order from the cart and checkout process, where a unique order number is generated and the admin interface is updated with the order details, payment is made and the user is redirected to the order page showing their listed items and confirmed that the payment has been received.
Level	User Goal
Description	The use case illustrates the process where the user/shopper proceeds with their order, the order received a unique order number for reference and the system is updated on the admin interface. The back end is updated with their order information.
Author	Kerrie Newman Ryan
Last Update:	July 2023
Status:	Completed
Use Cases:	Cart Checkout, Generate Order Number, Back end update
Primary Actor:	User/Customer/Shopper
Secondary Actors	Database, Admin interface
	Flow Description:
Preconditions:	The user has items in their cart and wish to proceed with the purchase
Assumptions:	The user is logged in and is ready to proceed to checkout
Trigger:	The user clicks on the cart icon and then "Proceed to Checkout" to continue with the purchase.
Success Post Condition:	The user is brought to the payment gateway to complete the order and purchase. After the payment has been received, redirected to the order details page showing all of the billing information, total cost and paid status
Failed Post Condition:	The user is brought back to their login page and asked to log in
Normal Flow:	**(User Proceeds to Checkout):** 1 The user clicks on the "Proceed to Checkout" button from the shopping cart. 2.The system presents the checkout page, detailing items, quantities, prices, total cost, and payment options. 3.The user confirms their shipping address and selects a payment method. 4. The user confirms the order details and clicks "Place Order." 5. The system processes the payment through the PayPal gateway.

	6.The pages shows “Please Wait” while processing the payment 7.Upon successful payment, the system generates a unique order number for the purchase. 8.A confirmation page is displayed to the user with the order details and the unique order number, the date of the order, payment method and transaction id. 9.The system updates the backend (e.g., database) with the order details, including items purchased, quantities, shipping address, user details, generated order number, the transaction id number 10.The user receives an email confirmation with the order details and the unique order number. 11. The order is available on the “My Order” for future reference. 12. The business also receives an email confirming they are have a new order that needs to be fulfilled
Alternative Flow:	**A1: Payment Declined** 1.After the user clicks "Place Order," the payment gateway declines the transaction. 2.The system notifies the user about the declined payment and prompts them to use a different payment method or check the provided payment details.
Exceptional Flow:	**Exceptional Flow: E1: System Error during Checkout** 1.Due to technical glitches, database retrieval issues, or connection problems with the payment gateway, the checkout process is interrupted. 2.The system displays an error message: "Unable to process your order at this time. Please try again later." 3.The user can decide to try again or continue shopping.
Termination	After successfully placing the order or encountering an error, the user can decide to continue shopping, view their order history, or leave the platform.
Post Situation	The system awaits further interactions or actions from the user
Overview	This use case provides a detailed blueprint of how the online marketplace system facilitates the checkout process, ensuring that users receive prompt feedback, a reference for their purchases, and confidence in the system's reliability. By generating a unique order number, the platform offers users a way to track and reference their

	transactions, enhancing trust and ensuring smooth post-purchase interactions.

2.2.1. Data Requirements

- **User Accounts:** User ID, Name, Email Address, Password (that has been encrypted using hashing), Address, Geolocation (latitude and longitude), Phone Number, Date Joined and Last Login Date.
- **User Roles:** Business, Customer and Admin(Staff)
- **Categories:** Category id, Category Name and Category Description
- **Product and Service Details:** Product ID , Product Name, Product Description, Price, Item Quantity, Product Image and Related Category
- **Order Data:** Orders, Order ID, Ordered Items, Payment Method, Transaction ID, Quantity ordered, Total price/cost of the order, Shipping Address
- **Payment Data:** Payment method, Payment Date, Amount Paid, Transaction Id and Payment Id

2.2.2. User Requirements

Customer Users/Shoppers:

- Users must be able to sign up and log in safely to the system.
- Users should have the option to explore and do product searches.
- Users must have the option to add items to carts and then check out and make payment securely.
- Users must be able to look up their previous orders and monitor the progress of their current orders.

Business Users:

- Business users must be able to log in safely
- Business Users must be able to update their profile and address to address to ordering
- Business users must have the options to manage, maintain and edit their categories and menu items through their dashboard
- Business users must have the option to manage their active orders, view their revenue and view their order history related to their business only.

Back End Admin (Communication Station Team)

- Has Admin access to all of the above and access to manage user data when needed.

2.2.3. Environmental Requirements

- The required software stack for the system is Python 3.12, Django 4.2, and PostgreSQL 15.
- To guarantee data integrity and availability, the server hosting the system should have regular backups and redundancy. This is managed through the PgAdmin system from PostGresSQL15
- Sensitive data must be stored and sent using secure encryption techniques, for example password hashing so that even the back end admins don't have access to the users passwords and information.
- A hosting environment that can scale as the user base grows and is ideal for multiple concurrent users. In this case, Heroku and Linode would be ideal for this.
- Integration capabilities with external services, including email notification systems or payment gateways.

2.2.4. Usability Requirements

- The system interface is user friendly and intuitive for both customer and business users providing a positive user experience with each user.
- The interface and system is responsive across a range of devices and mobile phones.
- Event and Error handling ensure that the user is notified of any errors occurring on the system and how to resolve them easily.
- Search and navigation tools that work well make it simple to find products and information.
- In order to give users a seamless experience, pages should load promptly and update the information retroactively for example cart and menu updates.
- Considerations for accessibility, such colour contrast and font legibility, are made to accommodate all users, including those with disabilities.

2.3.Design & Architecture

1.System Overview:

This online business and order management systems "Communication Station" was designed to facilitate customers browse products and services, adding them to their cart and purchase. This also allows business to add and manage their revenue, services and outgoing menu items.

The communication stations backend is built using Django and Python, with the front using HTML, Bootstrap and JavaScript with all of the data stored securely using PostgreSQL database.

Frontend: Client-side interface where users and businesses interact with the system.

Backend: Server-side where main processing occurs.

Database: Storage for business listings, products, user accounts, and order data.

2. Backend Architecture (Django with Python)

Django Model-View-Controller (MVC) Pattern:

Model: Represents the database schema and data structure. We use Django's ORM to define models such as User, Business, Order, and Cart. Ensuring to make migrations each time the models have been updated.

View: Represents the logic to render data to the users. For instance, when a user wants to view all available men items, the "marketplace" retrieves the data from the "Category" and "menuItem" model and sends it to the frontend. This is managed in the "views.py" files for the respective folders.

Controller: In Django, this is referred to as the URL dispatcher. It directs web requests to the appropriate view based on the URL endpoint.

API Endpoints: Using Django Rest Framework, various API endpoints such as /menuitems, /cart, /order, etc., are exposed for the frontend to interact with the backend. This is all managed in the "urls.py" files

Business Management: Logic to add, update, or remove business listings.

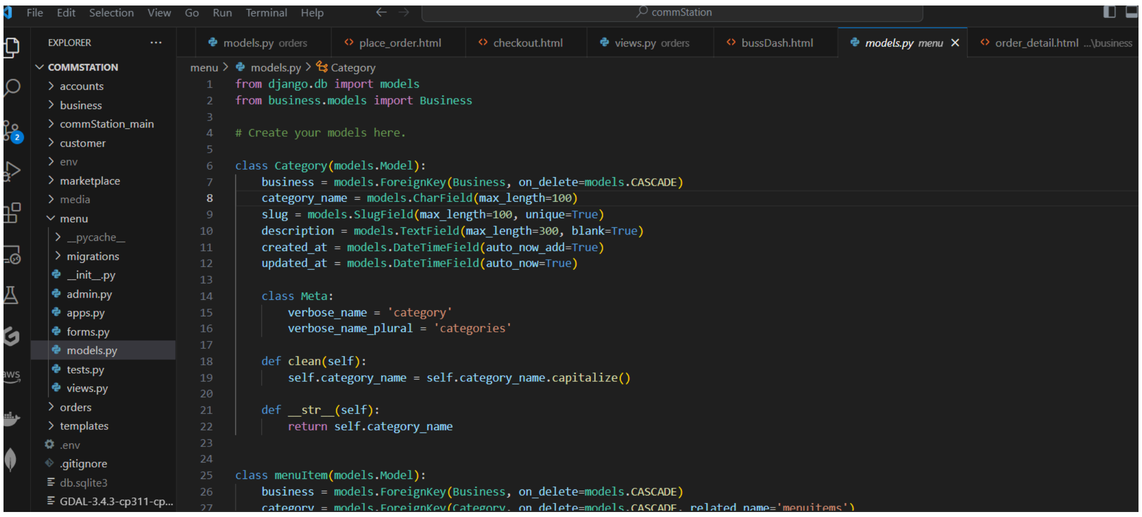
Product Management: Logic to add, update, or remove products for a business.

Order Processing: Logic to handle product orders, order status, and notifications.

Search Functionality: Efficient searching mechanism for products and businesses.

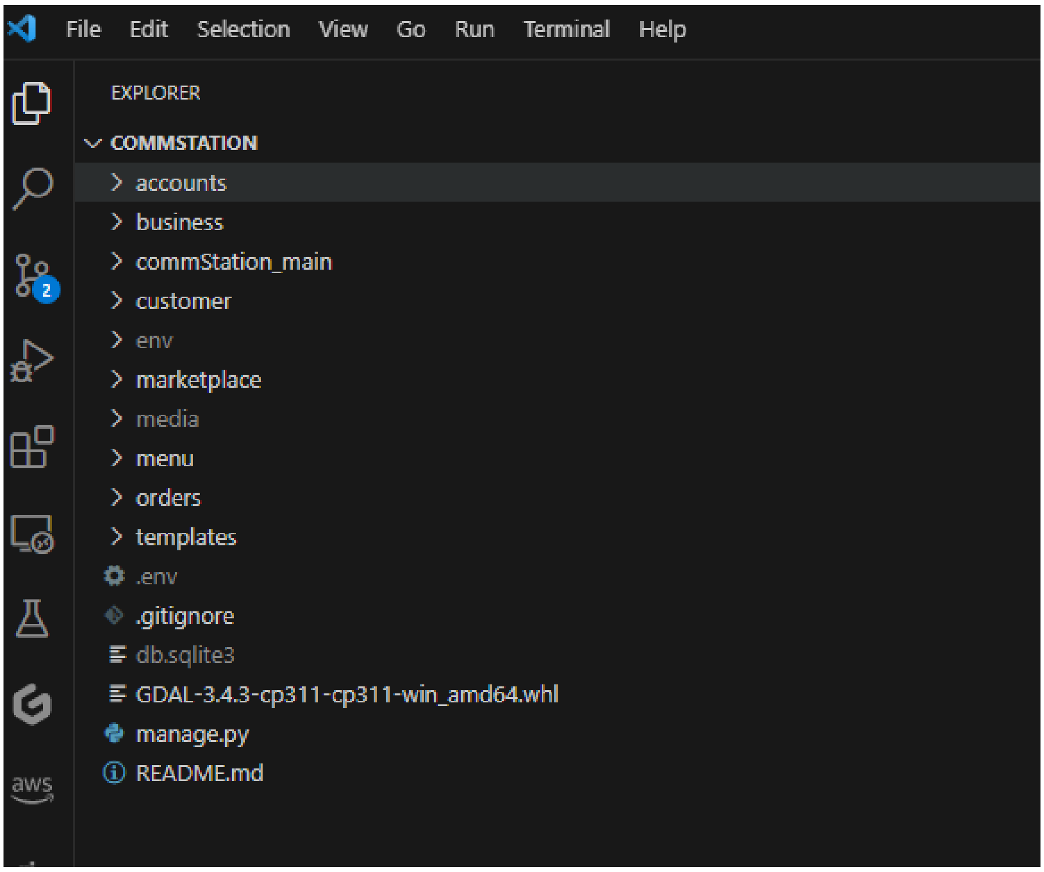
Payment Integration: PayPal SDK/API integration for payment processing.

GDAL Integration: Geographic libraries for location-based features, like finding businesses near a user.



```
from django.db import models
from business.models import Business

# Create your models here.

class Category(models.Model):
    business = models.ForeignKey(Business, on_delete=models.CASCADE)
    category_name = models.CharField(max_length=100)
    slug = models.SlugField(max_length=100, unique=True)
    description = models.TextField(max_length=300, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        verbose_name = 'category'
        verbose_name_plural = 'categories'

    def clean(self):
        self.category_name = self.category_name.capitalize()

    def __str__(self):
        return self.category_name


class menuItem(models.Model):
    business = models.ForeignKey(Business, on_delete=models.CASCADE)
    category = models.ForeignKey(Category, on_delete=models.CASCADE, related_name='menuitems')
```

Showing backend MVC in commStation folder



```
COMMSTATION
  > accounts
  > business
  > commStation_main
  > customer
  > env
  > marketplace
  > media
  > menu
  > orders
  > templates
  .env
  .gitignore
  db.sqlite3
  GDAL-3.4.3-cp311-cp311-win_amd64.whl
  manage.py
  README.md
```

Shows folder layout from VS code

2. Design & Components:

1.Frontend Components:

User Interface (UI): Web-based, developed using HTML/CSS/JS and Bootstrap

Cart System: Allows users to add products and proceed to checkout.

Users: Contains information about registered users.

Items: Stores product details such as name, price, description, and image.

Cart: Maintains the association between users and products they wish to purchase.

Orders: Logs finalised orders, associating a user with purchased products.

2. **Main Algorithms:**

Search Algorithm: Use of an inverted index for efficient text-based searching of products and businesses. Trie data structure can be employed for predictive search functionality.

Geolocation-based Search: Using spatial indices in databases (R-tree) to quickly find businesses/products near a user.

Payment Processing: Standard payment gateway integration process using OAuth for secure token-based communication with PayPal.

4. **Relationships:**

Users to Items: Many-to-Many, managed via the Cart table.

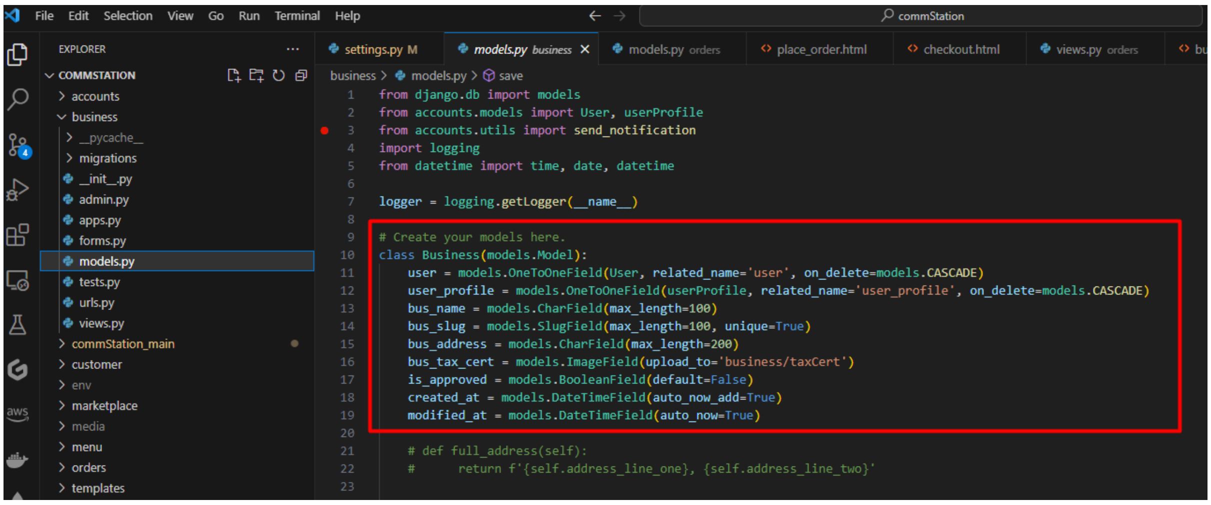
Users to Orders: One-to-Many, indicating that a user can place multiple orders.

Indexing: Given the expected volume of data, we use B-tree indexing on frequently searched columns, such as menuItem in the menu table, to accelerate read operations.

2.4. Implementation

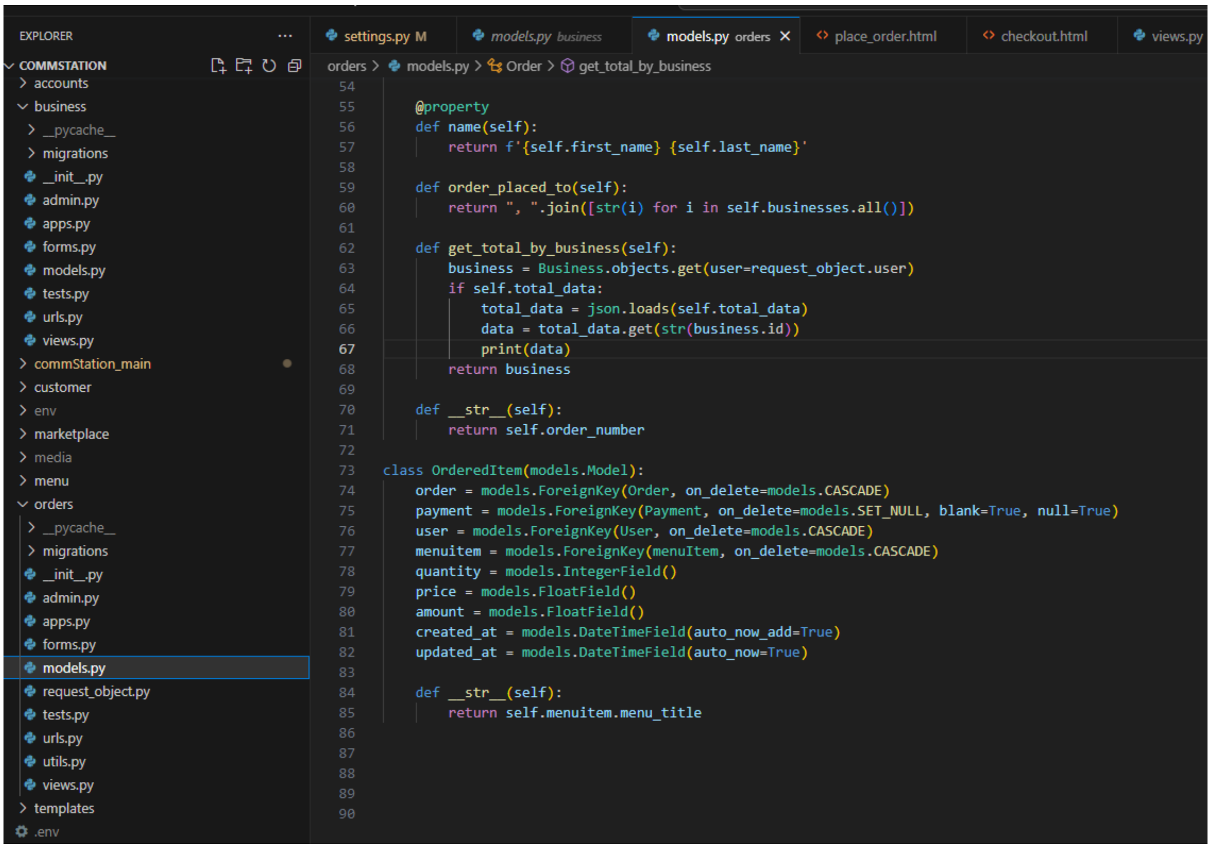
Backend Implementation (Django with Python)

Business Model shows the User information required for the businesses on the platform:

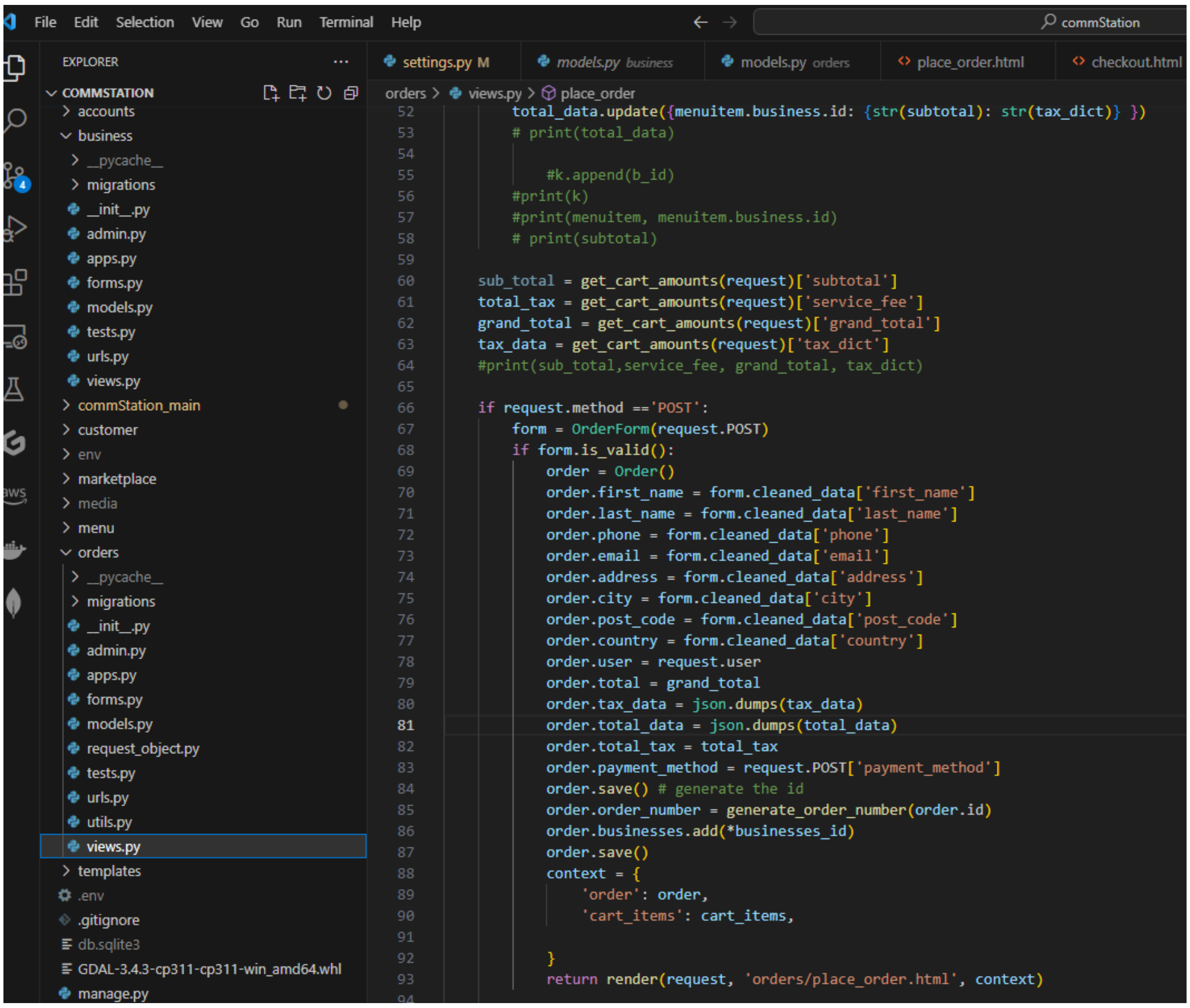


```python
from django.db import models
from accounts.models import User, userProfile
from accounts.utils import send_notification
import logging
from datetime import time, date, datetime

logger = logging.getLogger(__name__)

# Create your models here.
class Business(models.Model):
    user = models.OneToOneField(User, related_name='user', on_delete=models.CASCADE)
    user_profile = models.OneToOneField(userProfile, related_name='user_profile', on_delete=models.CASCADE)
    bus_name = models.CharField(max_length=100)
    bus_slug = models.SlugField(max_length=100, unique=True)
    bus_address = models.CharField(max_length=200)
    bus_tax_cert = models.ImageField(upload_to='business/taxCert')
    is_approved = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    modified_at = models.DateTimeField(auto_now=True)

    # def full_address(self):
    #      return f'{self.address_line_one}, {self.address_line_two}'
```

Ordered Item Model to handle the orders models:

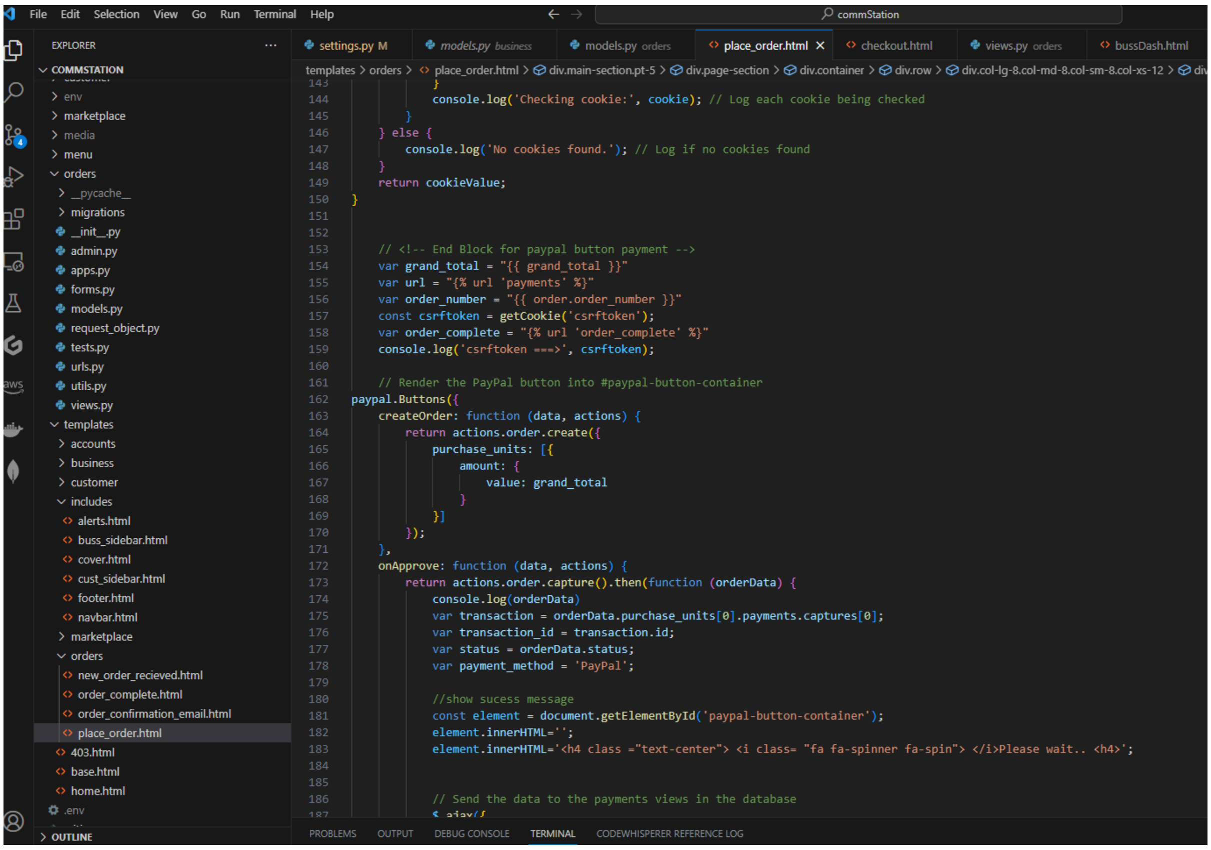


```python
    @property
    def name(self):
        return f'{self.first_name} {self.last_name}'

    def order_placed_to(self):
        return ", ".join([str(i) for i in self.businesses.all()])

    def get_total_by_business(self):
        business = Business.objects.get(user=request_object.user)
        if self.total_data:
            total_data = json.loads(self.total_data)
            data = total_data.get(str(business.id))
            print(data)
        return business

    def __str__(self):
        return self.order_number

class OrderedItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    payment = models.ForeignKey(Payment, on_delete=models.SET_NULL, blank=True, null=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    menuitem = models.ForeignKey(menuItem, on_delete=models.CASCADE)
    quantity = models.IntegerField()
    price = models.FloatField()
    amount = models.FloatField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.menuitem.menu_title
```



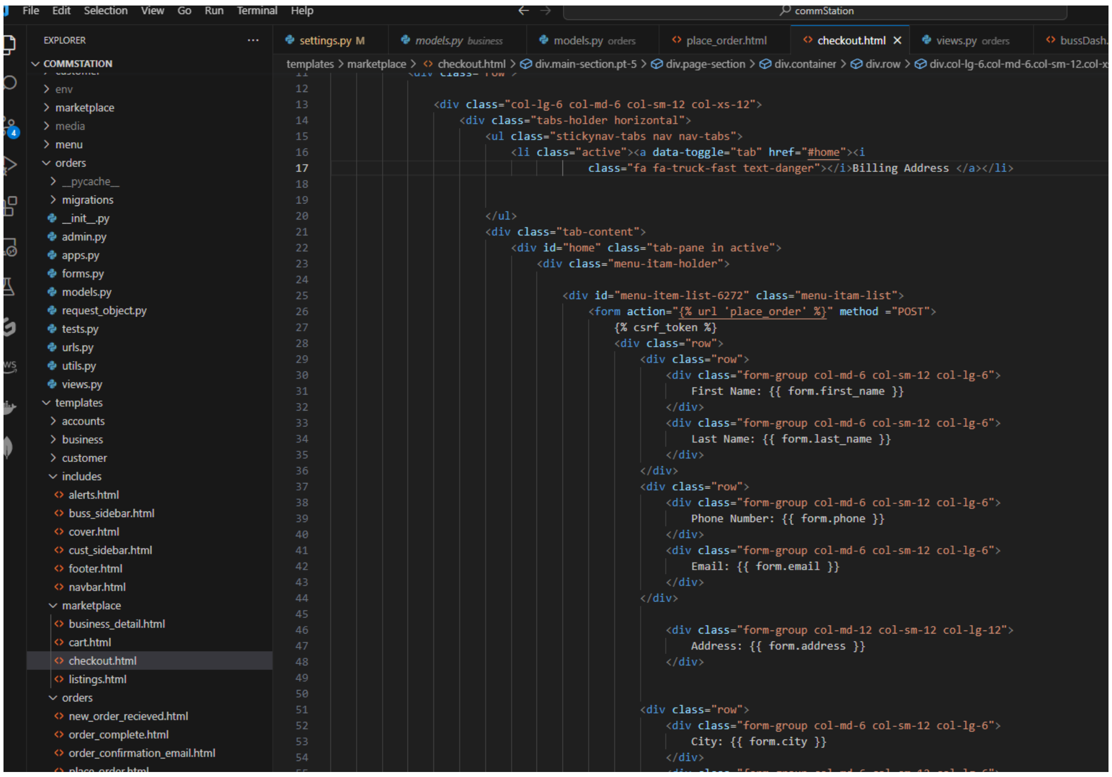
```
        total_data.update({menuitem.business.id: {str(subtotal): str(tax_dict)} })
        # print(total_data)

            #k.append(b_id)
        #print(k)
        #print(menuitem, menuitem.business.id)
        # print(subtotal)

    sub_total = get_cart_amounts(request)['subtotal']
    total_tax = get_cart_amounts(request)['service_fee']
    grand_total = get_cart_amounts(request)['grand_total']
    tax_data = get_cart_amounts(request)['tax_dict']
    #print(sub_total,service_fee, grand_total, tax_dict)

    if request.method =='POST':
        form = OrderForm(request.POST)
        if form.is_valid():
            order = Order()
            order.first_name = form.cleaned_data['first_name']
            order.last_name = form.cleaned_data['last_name']
            order.phone = form.cleaned_data['phone']
            order.email = form.cleaned_data['email']
            order.address = form.cleaned_data['address']
            order.city = form.cleaned_data['city']
            order.post_code = form.cleaned_data['post_code']
            order.country = form.cleaned_data['country']
            order.user = request.user
            order.total = grand_total
            order.tax_data = json.dumps(tax_data)
            order.total_data = json.dumps(total_data)
            order.total_tax = total_tax
            order.payment_method = request.POST['payment_method']
            order.save() # generate the id
            order.order_number = generate_order_number(order.id)
            order.businesses.add(*businesses_id)
            order.save()
            context = {
                'order': order,
                'cart_items': cart_items,

            }
            return render(request, 'orders/place_order.html', context)
```

The views.py file that handles the order from the cart request and add the related user data to the billing address.



```
            console.log('Checking cookie:', cookie); // Log each cookie being checked
        }
    } else {
        console.log('No cookies found.'); // Log if no cookies found
    }
    return cookieValue;
}


    // <!-- End Block for paypal button payment -->
    var grand_total = "{{ grand_total }}"
    var url = "{% url 'payments' %}"
    var order_number = "{{ order.order_number }}"
    const csrftoken = getCookie('csrftoken');
    var order_complete = "{% url 'order_complete' %}"
    console.log('csrftoken ===>', csrftoken);

    // Render the PayPal button into #paypal-button-container
paypal.Buttons({
    createOrder: function (data, actions) {
        return actions.order.create({
            purchase_units: [{
                amount: {
                    value: grand_total
                }
            }]
        });
    },
    onApprove: function (data, actions) {
        return actions.order.capture().then(function (orderData) {
            console.log(orderData)
            var transaction = orderData.purchase_units[0].payments.captures[0];
            var transaction_id = transaction.id;
            var status = orderData.status;
            var payment_method = 'PayPal';

            //show sucess message
            const element = document.getElementById('paypal-button-container');
            element.innerHTML='';
            element.innerHTML='<h4 class ="text-center"> <i class= "fa fa-spinner fa-spin"> </i>Please wait.. <h4>';


            // Send the data to the payments views in the database
```

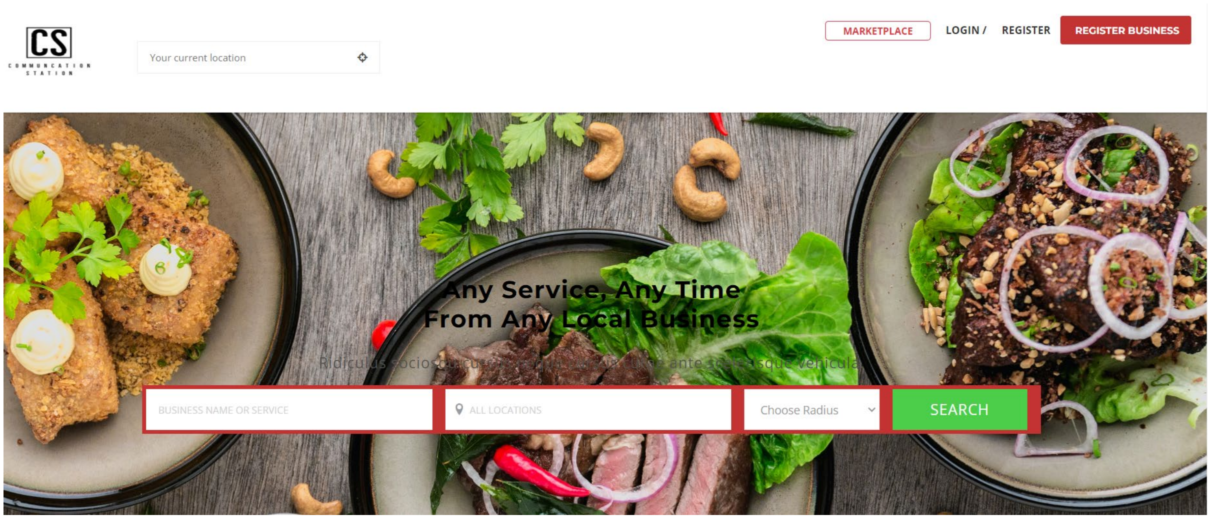
JavaScript implementation for the PayPal integration accepting payments



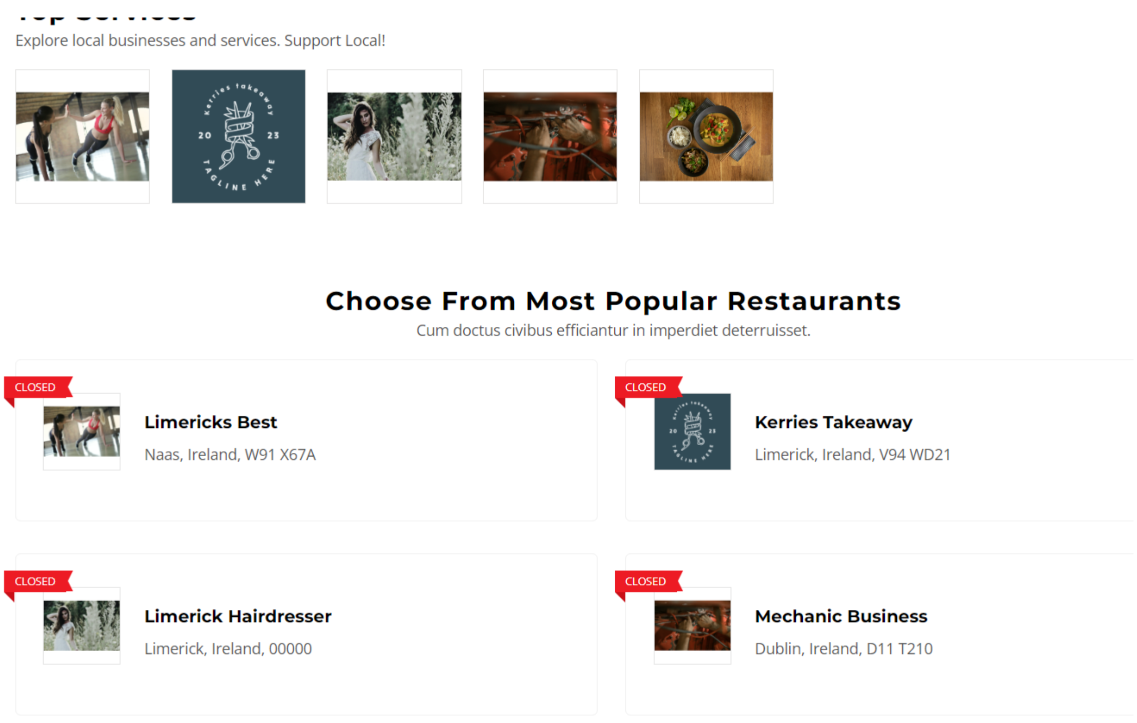
```

<div class="col-lg-6 col-md-6 col-sm-12 col-xs-12">
    <div class="tabs-holder horizontal">
        <ul class="stickynav-tabs nav nav-tabs">
            <li class="active"><a data-toggle="tab" href="#home"><i
                        class="fa fa-truck-fast text-danger"></i>Billing Address </a></li>


        </ul>
        <div class="tab-content">
            <div id="home" class="tab-pane in active">
                <div class="menu-itam-holder">

                    <div id="menu-item-list-6272" class="menu-itam-list">
                        <form action="{% url 'place_order' %}" method ="POST">
                            {% csrf_token %}
                            <div class="row">
                                <div class="row">
                                    <div class="form-group col-md-6 col-sm-12 col-lg-6">
                                        First Name: {{ form.first_name }}
                                    </div>
                                    <div class="form-group col-md-6 col-sm-12 col-lg-6">
                                        Last Name: {{ form.last_name }}
                                    </div>
                                </div>
                                <div class="row">
                                    <div class="form-group col-md-6 col-sm-12 col-lg-6">
                                        Phone Number: {{ form.phone }}
                                    </div>
                                    <div class="form-group col-md-6 col-sm-12 col-lg-6">
                                        Email: {{ form.email }}
                                    </div>
                                </div>

                                    <div class="form-group col-md-12 col-sm-12 col-lg-12">
                                        Address: {{ form.address }}
                                    </div>


                                <div class="row">
                                    <div class="form-group col-md-6 col-sm-12 col-lg-6">
                                        City: {{ form.city }}
                                    </div>
```

Checkout with the forms.py from the accounts folder

Graphical User Interface (GUI)



Home page with search function


Explore local businesses and services. Support Local!

Choose From Most Popular Restaurants

Cum doctus civibus efficiantur in imperdiet deterruisset.

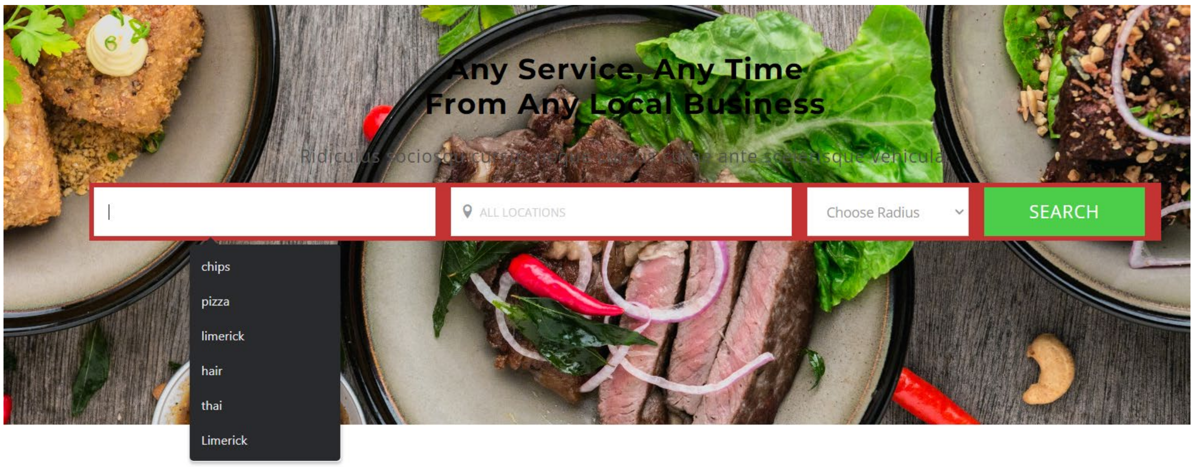
CLOSED
Limericks Best
Naas, Ireland, W91 X67A

CLOSED
Kerries Takeaway
Limerick, Ireland, V94 WD21

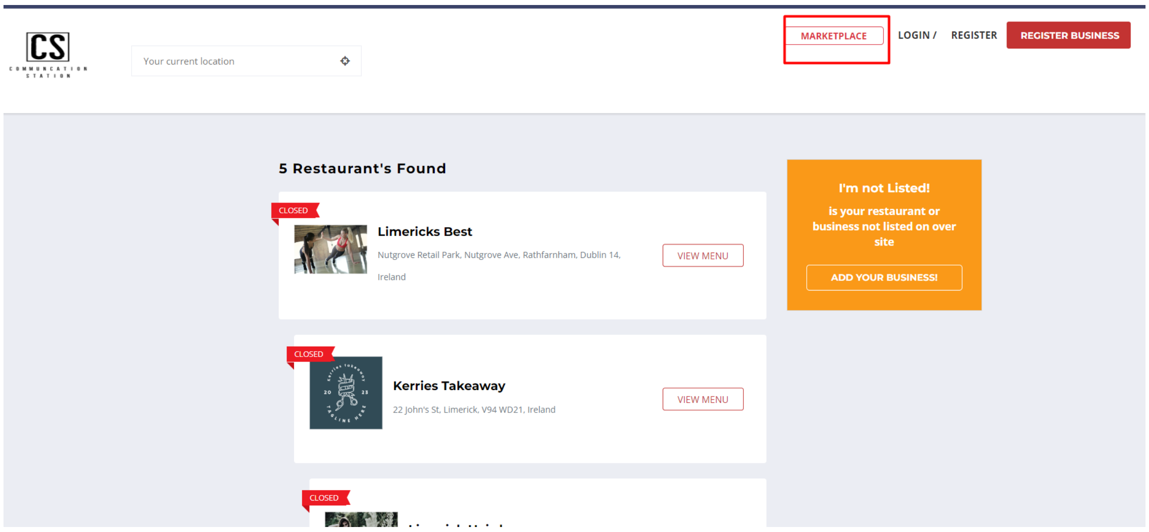
CLOSED
Limerick Hairdresser
Limerick, Ireland, 00000

CLOSED
Mechanic Business
Dublin, Ireland, D11 T210

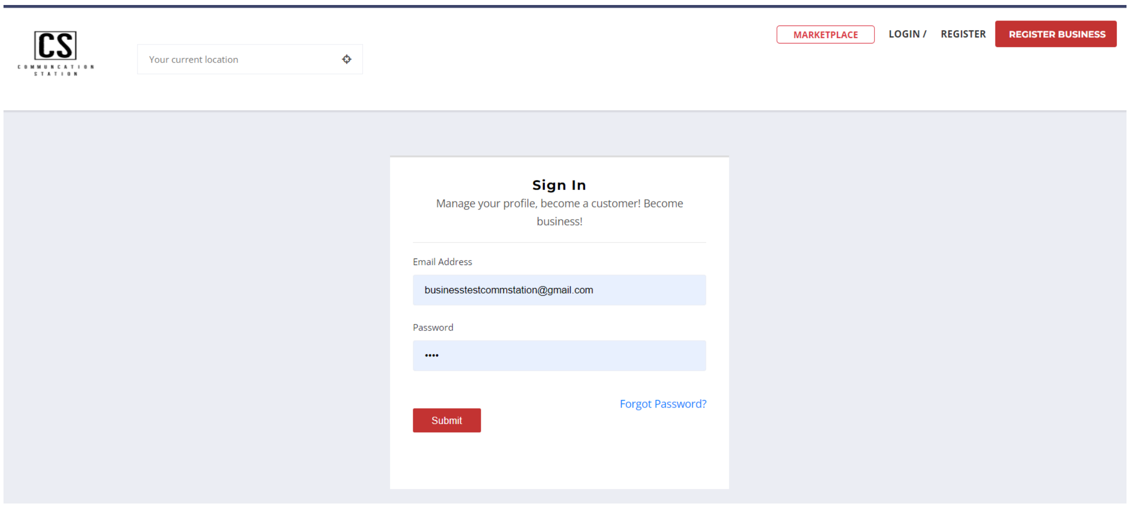
Available business and restaurants on the home page



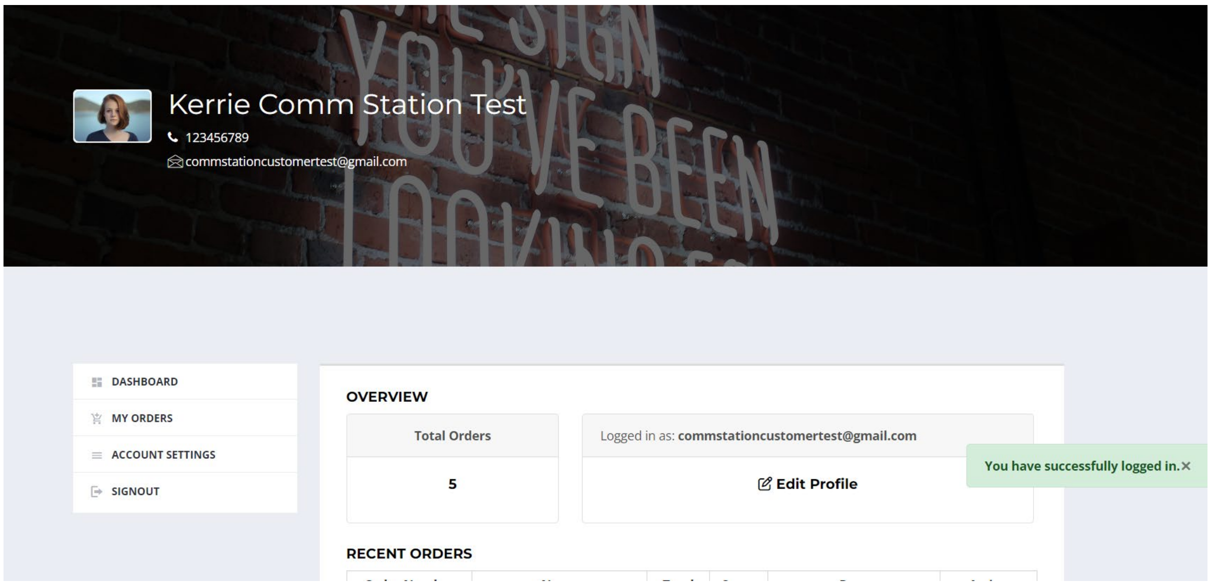
Search function on the home page location with Google Maps API and GDAL Geo location.



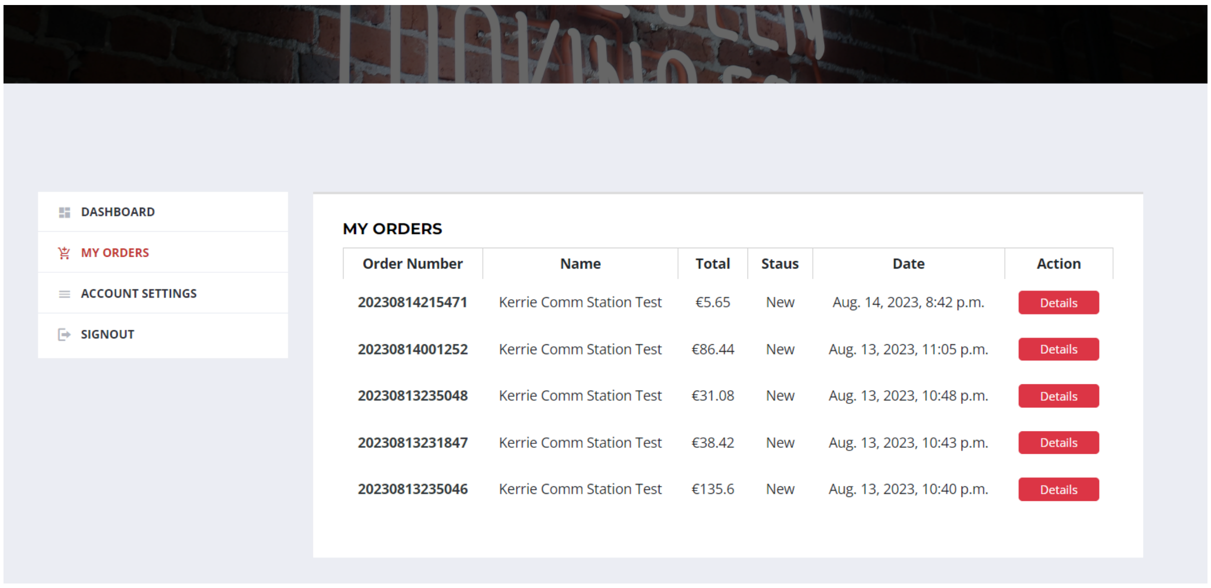
Marketplace showing available and approved restaurants



Login page for customers/users



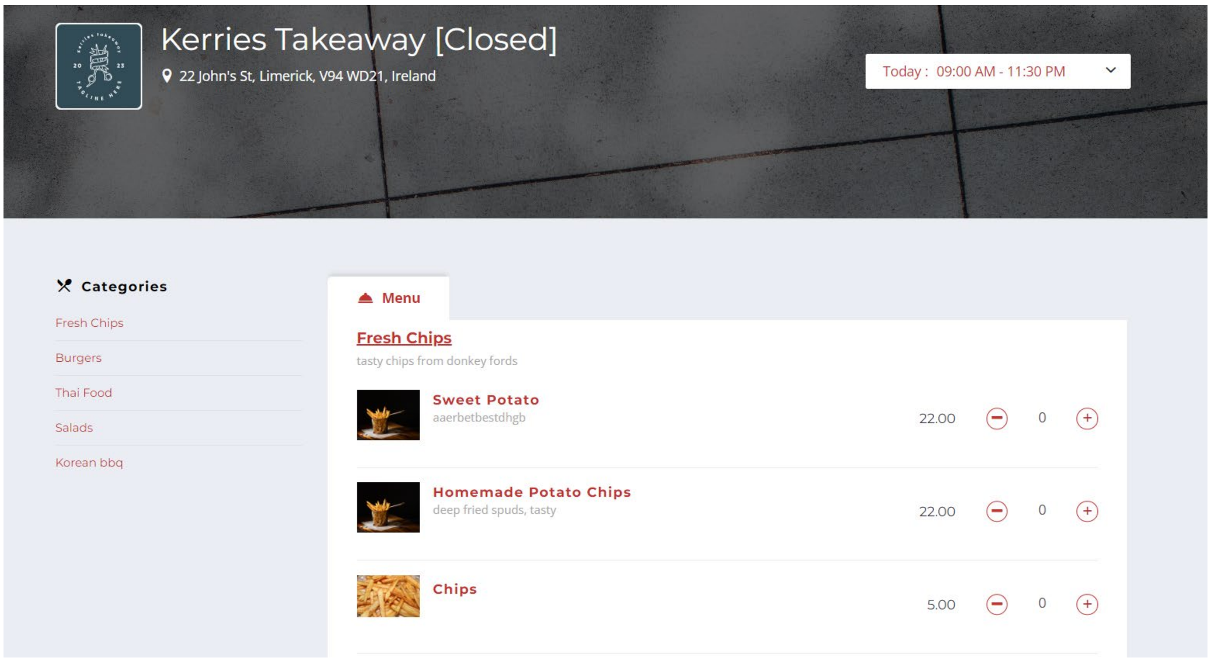
Customer dashboard



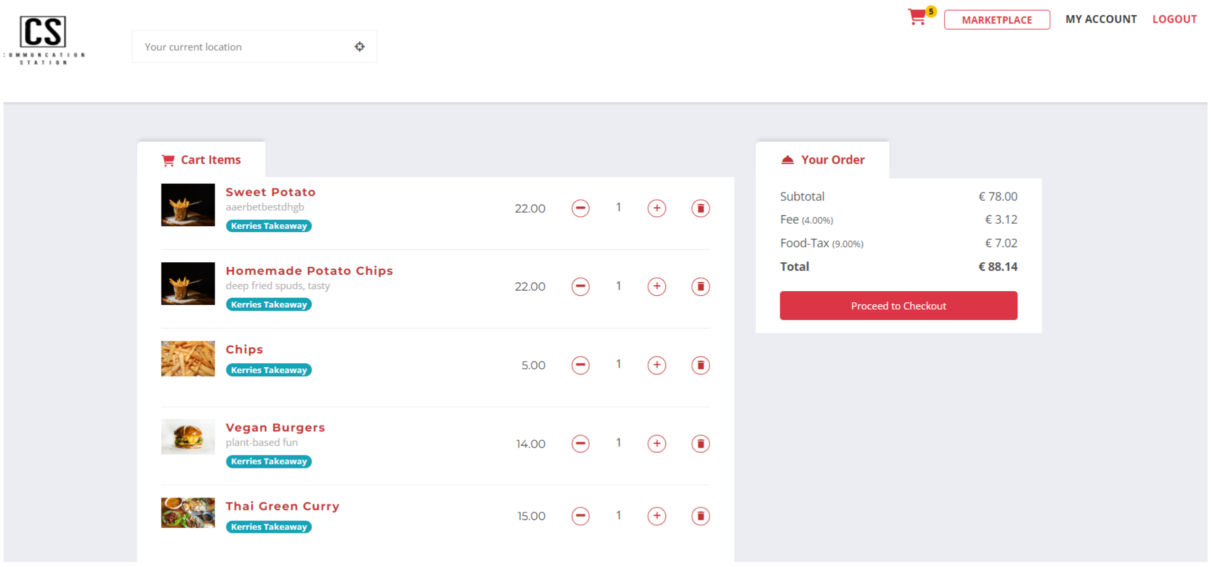
MY ORDERS

Order Number	Name	Total	Staus	Date	Action
20230814215471	Kerrie Comm Station Test	€5.65	New	Aug. 14, 2023, 8:42 p.m.	Details
20230814001252	Kerrie Comm Station Test	€86.44	New	Aug. 13, 2023, 11:05 p.m.	Details
20230813235048	Kerrie Comm Station Test	€31.08	New	Aug. 13, 2023, 10:48 p.m.	Details
20230813231847	Kerrie Comm Station Test	€38.42	New	Aug. 13, 2023, 10:43 p.m.	Details
20230813235046	Kerrie Comm Station Test	€135.6	New	Aug. 13, 2023, 10:40 p.m.	Details

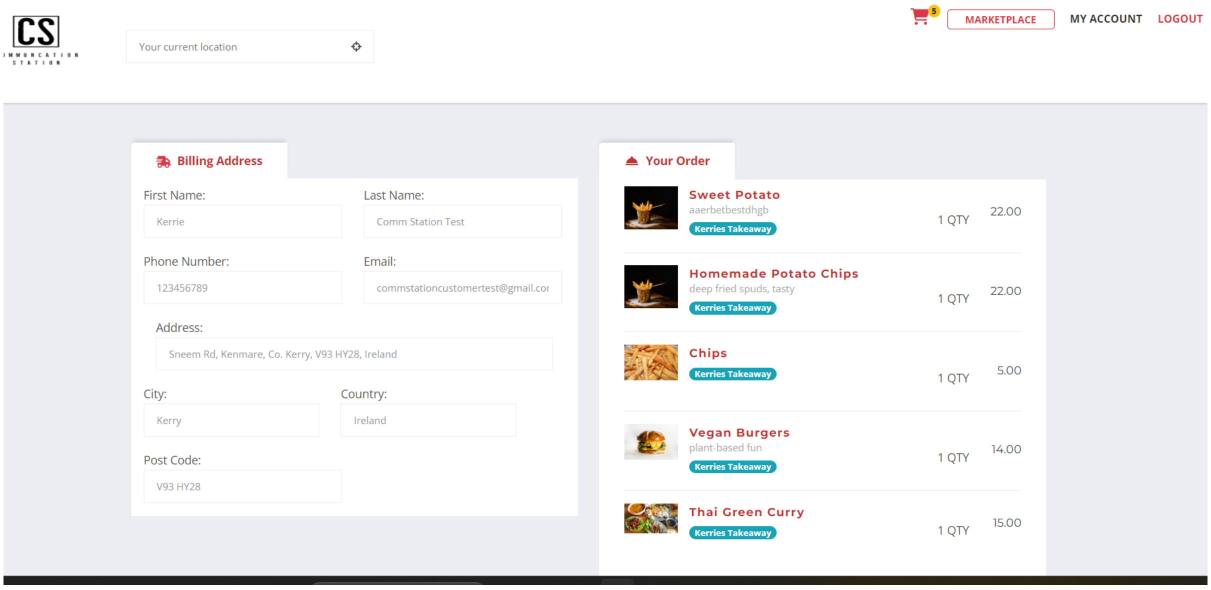
Order history



Order page with menu items



Cart Items and total amount with tax and service included



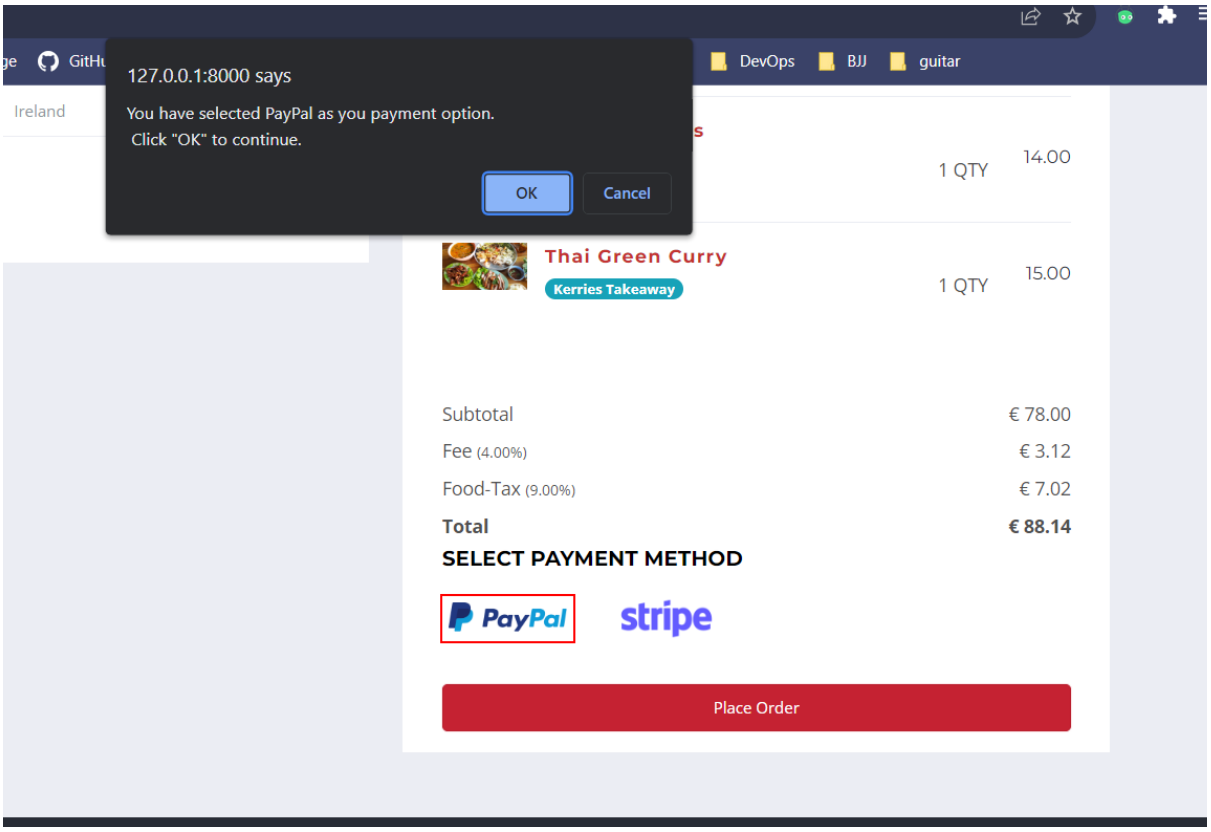
Billing Address

First Name: Kerrie
Last Name: Comm Station Test
Phone Number: 123456789
Email: commstationcustomertest@gmail.co
Address: Sneem Rd, Kenmare, Co. Kerry, V93 HY28, Ireland
City: Kerry
Country: Ireland
Post Code: V93 HY28

Your Order

Sweet Potato aaerbetbestdhgb Kerries Takeaway	1 QTY	22.00
Homemade Potato Chips deep fried spuds, tasty Kerries Takeaway	1 QTY	22.00
Chips Kerries Takeaway	1 QTY	5.00
Vegan Burgers plant-based fun Kerries Takeaway	1 QTY	14.00
Thai Green Curry Kerries Takeaway	1 QTY	15.00

Checkout with billing address, total amount owed and quantity



127.0.0.1:8000 says

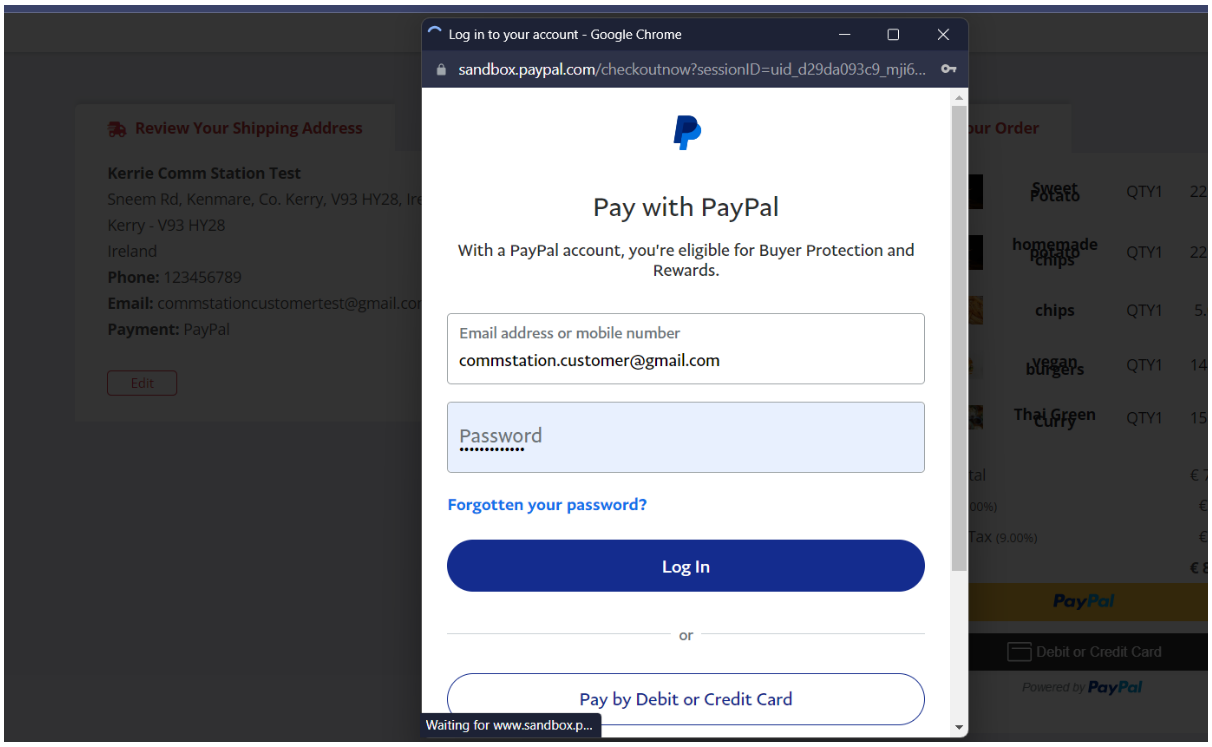
You have selected PayPal as you payment option.
Click "OK" to continue.

	1 QTY	14.00
Thai Green Curry Kerries Takeaway	1 QTY	15.00

Subtotal	€ 78.00
Fee (4.00%)	€ 3.12
Food-Tax (9.00%)	€ 7.02
Total	**€ 88.14**

SELECT PAYMENT METHOD

Payment option with PayPal


Log in to your account - Google Chrome
sandbox.paypal.com/checkoutnow?sessionID=uid_d29da093c9_mji6...
Review Your Shipping Address
Edit
Log In
or
Pay by Debit or Credit Card
Waiting for www.sandbox.p...
Debit or Credit Card
Powered by PayPal

Kerrie Comm Station Test
Sneem Rd, Kenmare, Co. Kerry, V93 HY28, Ire
Kerry - V93 HY28
Ireland
Phone: 123456789
Email: commstationcustomertest@gmail.co
Payment: PayPal

Pay with PayPal

With a PayPal account, you're eligible for Buyer Protection and Rewards.

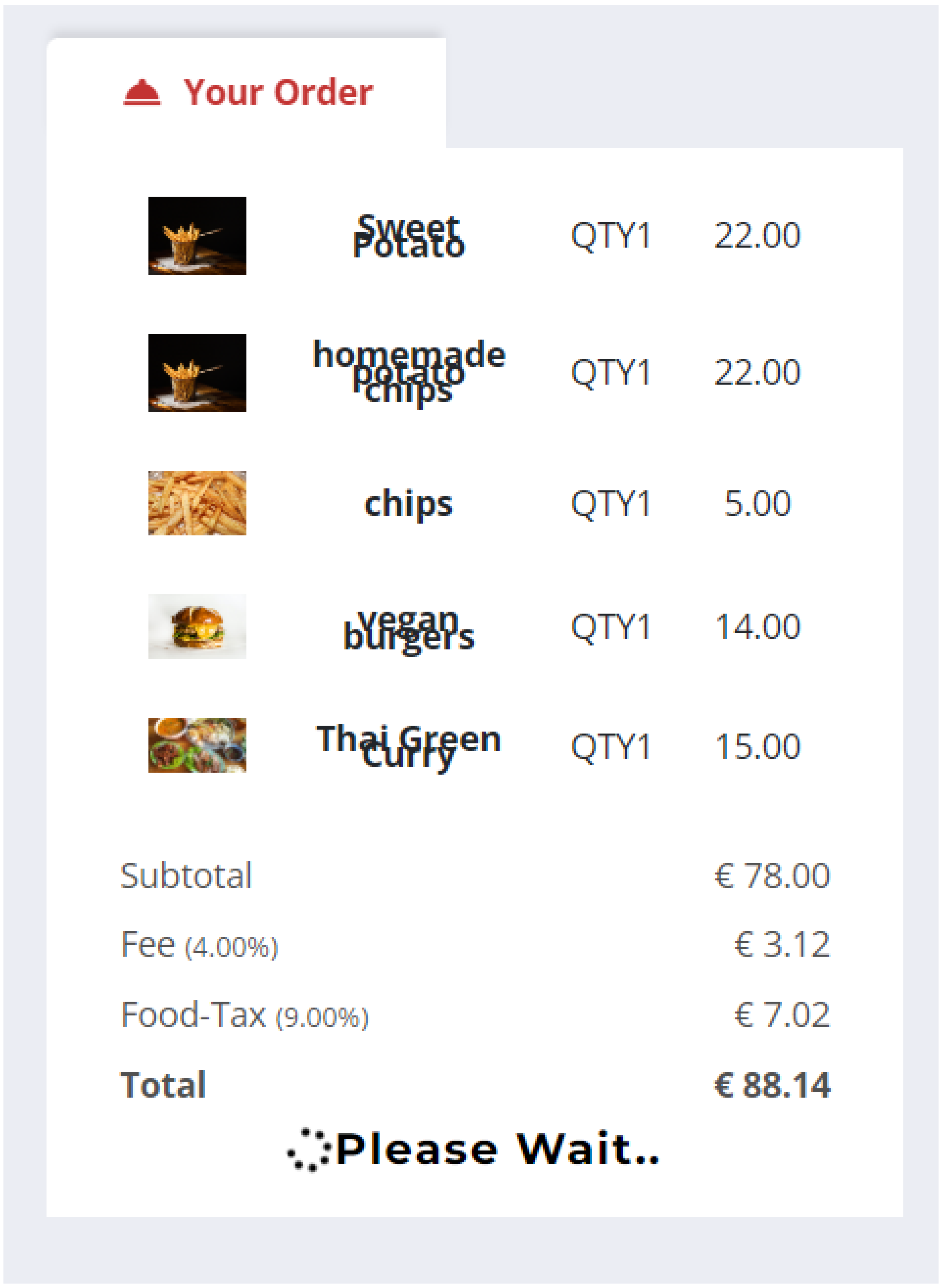
Email address or mobile number
commstation.customer@gmail.com

Password

Forgotten your password?

Sweet Potato	QTY1	22
homemade potato chips	QTY1	22
chips	QTY1	5.
vegan burgers	QTY1	14
Thai Green Curry	QTY1	15

PayPal integration operating

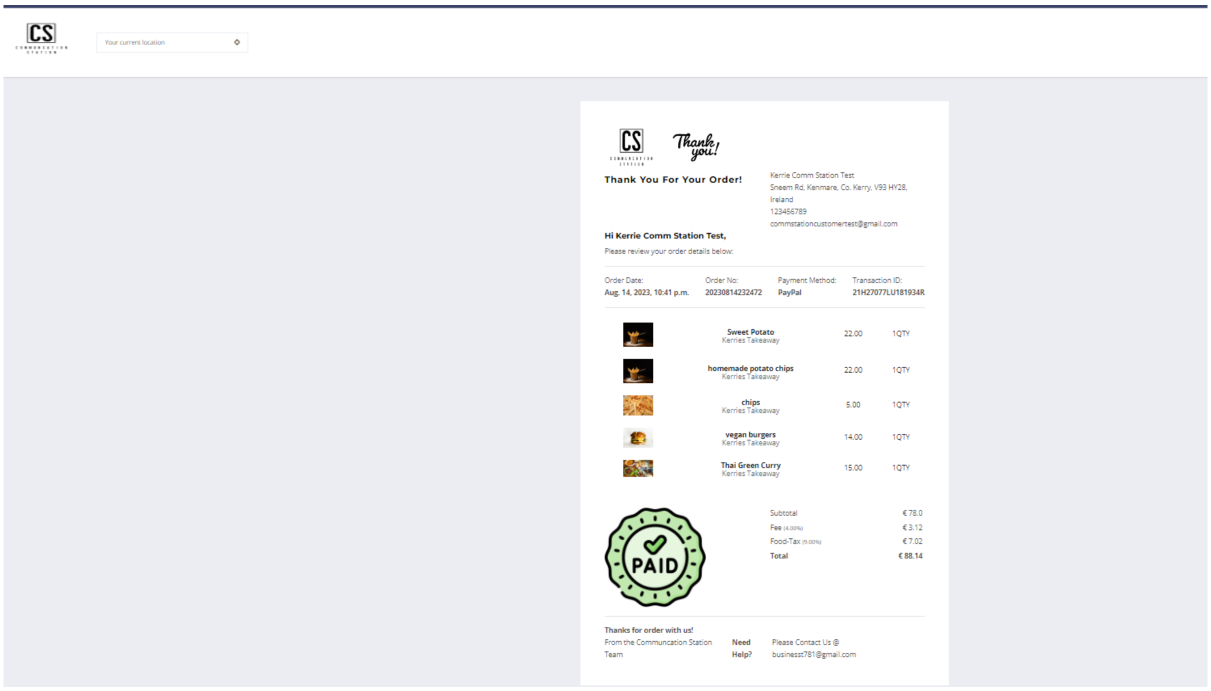

Your Order

	Sweet Potato	QTY1	22.00
	homemade potato chips	QTY1	22.00
	chips	QTY1	5.00
	vegan burgers	QTY1	14.00
	Thai Green Curry	QTY1	15.00

Subtotal	€ 78.00
Fee (4.00%)	€ 3.12
Food-Tax (9.00%)	€ 7.02
Total	**€ 88.14**

Please Wait..

Please wait while payment is processing



Thank You For Your Order!

Kerrie Comm Station Test
Sneem Rd, Kenmare, Co. Kerry, V93 HY28, Ireland
123456789
commstationcustomertest@gmail.com

Hi Kerrie Comm Station Test,

Please review your order details below:

Order Date:	Order No:	Payment Method:	Transaction ID:
Aug. 14, 2023, 10:41 p.m.	**20230814232472**	**PayPal**	**21H27077LU181934R**

Sweet Potato Kerries Takeaway	22.00	1QTY
homemade potato chips Kerries Takeaway	22.00	1QTY
chips Kerries Takeaway	5.00	1QTY
vegan burgers Kerries Takeaway	14.00	1QTY
Thai Green Curry Kerries Takeaway	15.00	1QTY

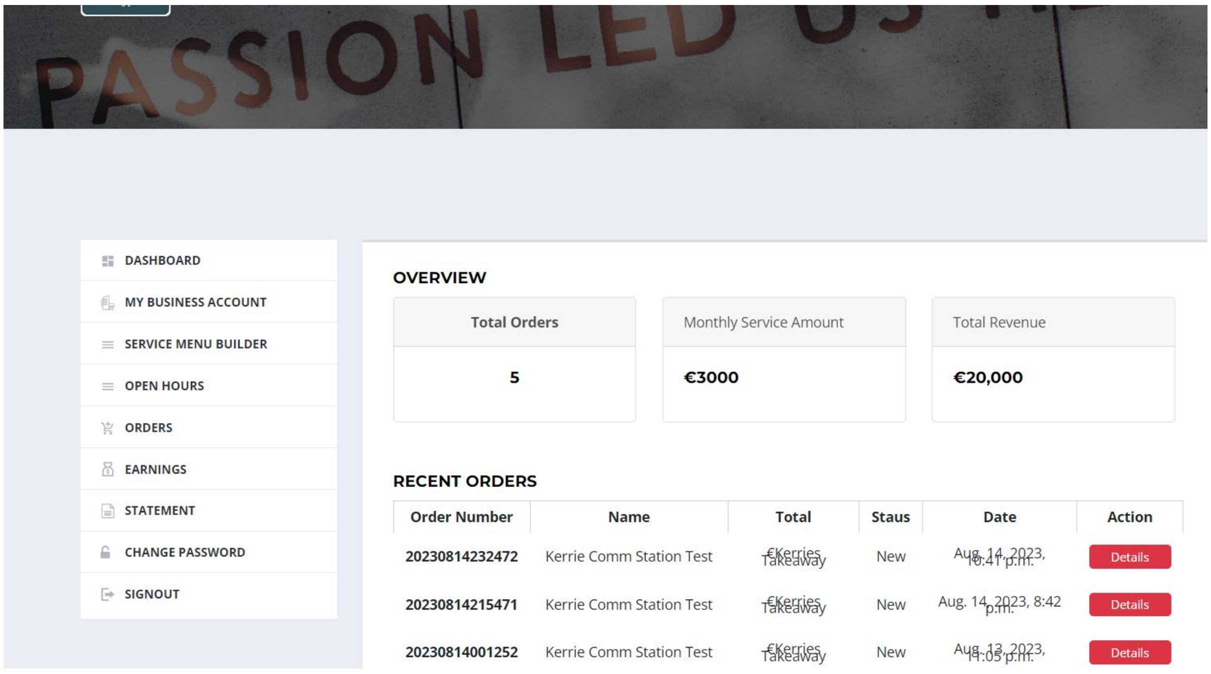
Subtotal	€ 78.0
Fee (4.00%)	€ 3.12
Food-Tax (9.00%)	€ 7.02
Total	**€ 88.14**

PAID

Thanks for order with us!
From the Communcation Station Team

Need Help? Please Contact Us @ businesst781@gmail.com

Order Invoice



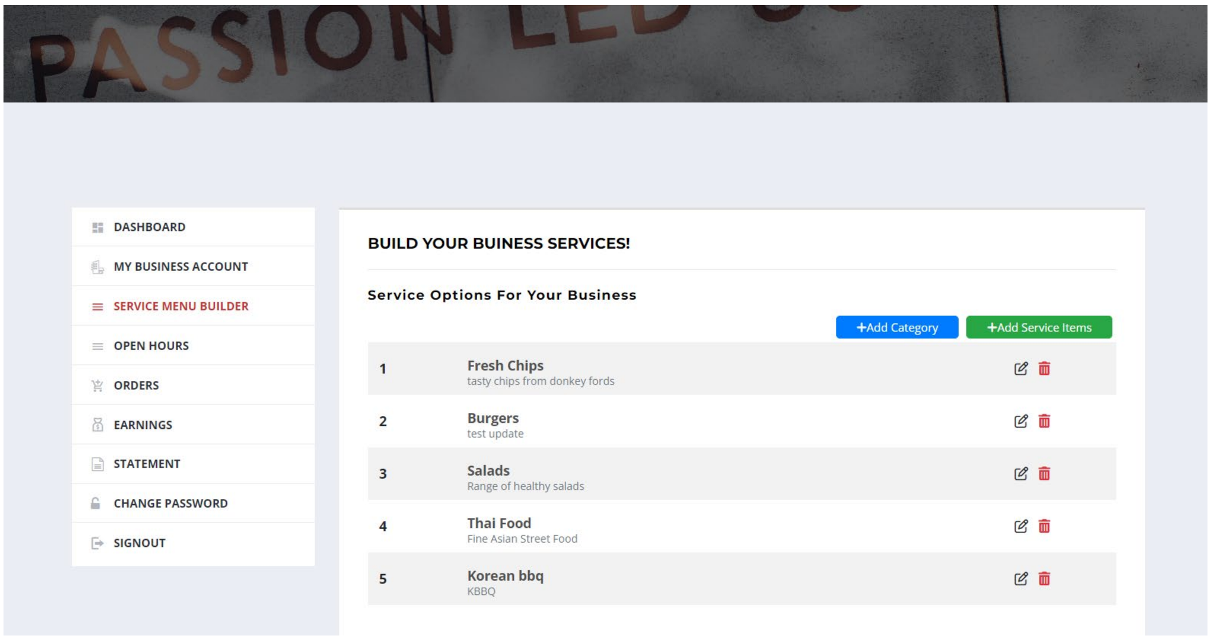
OVERVIEW

Total Orders	Monthly Service Amount	Total Revenue
5	€3000	€20,000

RECENT ORDERS

Order Number	Name	Total	Staus	Date	Action
20230814232472	Kerrie Comm Station Test	€Kerries Takeaway	New	Aug. 14, 2023, 10:41 p.m.	Details
20230814215471	Kerrie Comm Station Test	€Kerries Takeaway	New	Aug. 14, 2023, 8:42 p.m.	Details
20230814001252	Kerrie Comm Station Test	€Kerries Takeaway	New	Aug. 13, 2023, 11:05 p.m.	Details

Business Dashboard

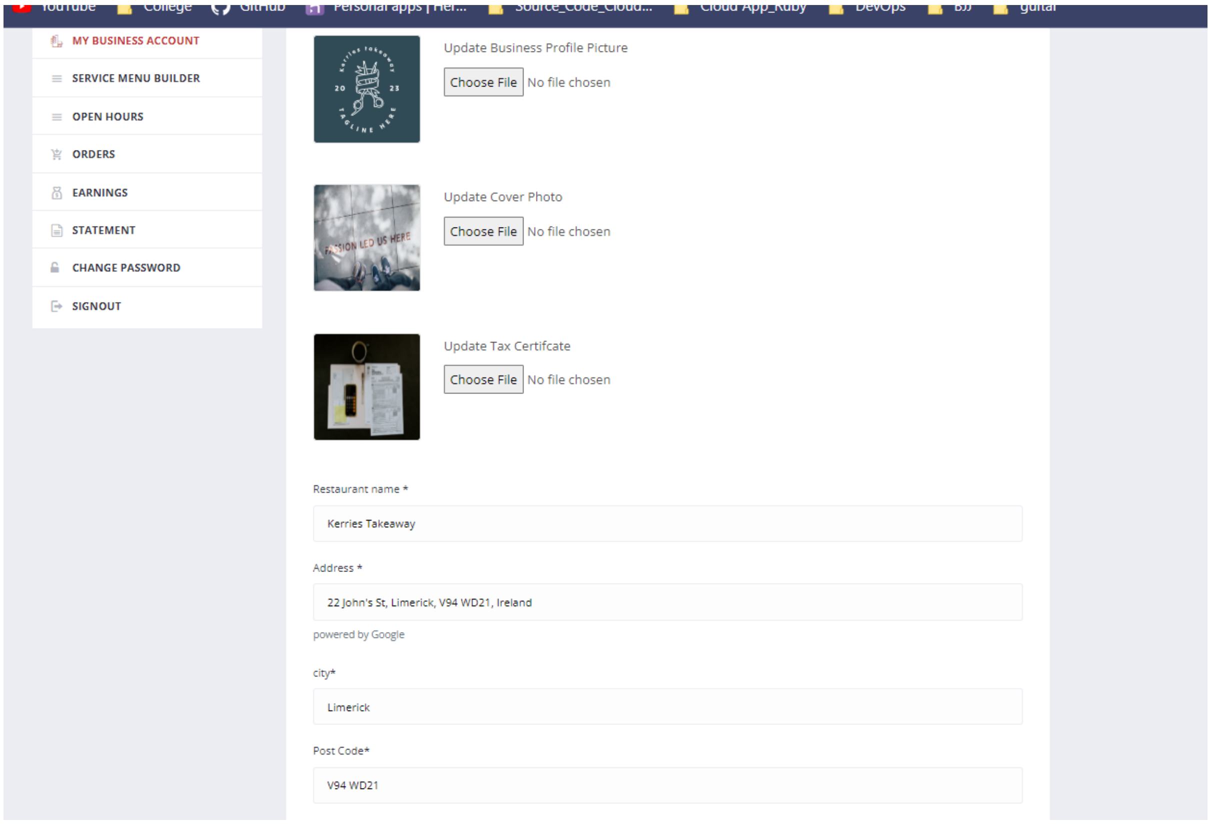


BUILD YOUR BUINESS SERVICES!

Service Options For Your Business

1	Fresh Chips	tasty chips from donkey fords
2	Burgers	test update
3	Salads	Range of healthy salads
4	Thai Food	Fine Asian Street Food
5	Korean bbq	KBBQ

Menu Builder



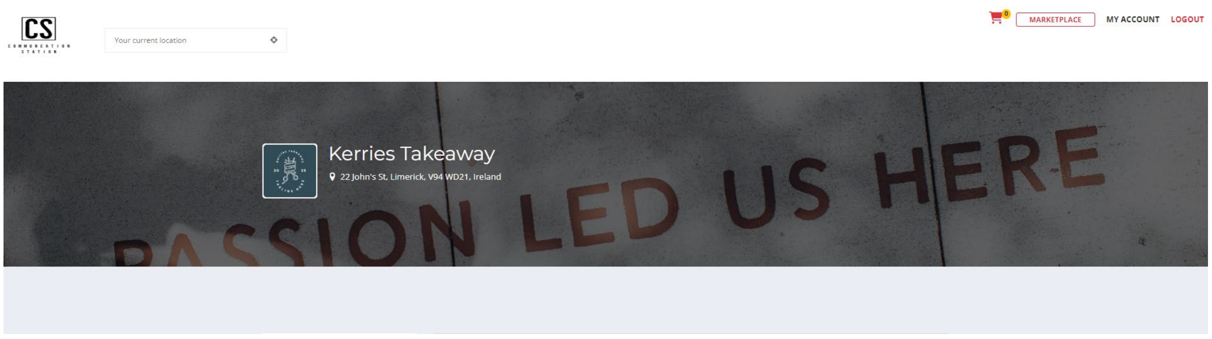
Restaurant name *
Kerries Takeaway

Address *
22 John's St, Limerick, V94 WD21, Ireland
powered by Google

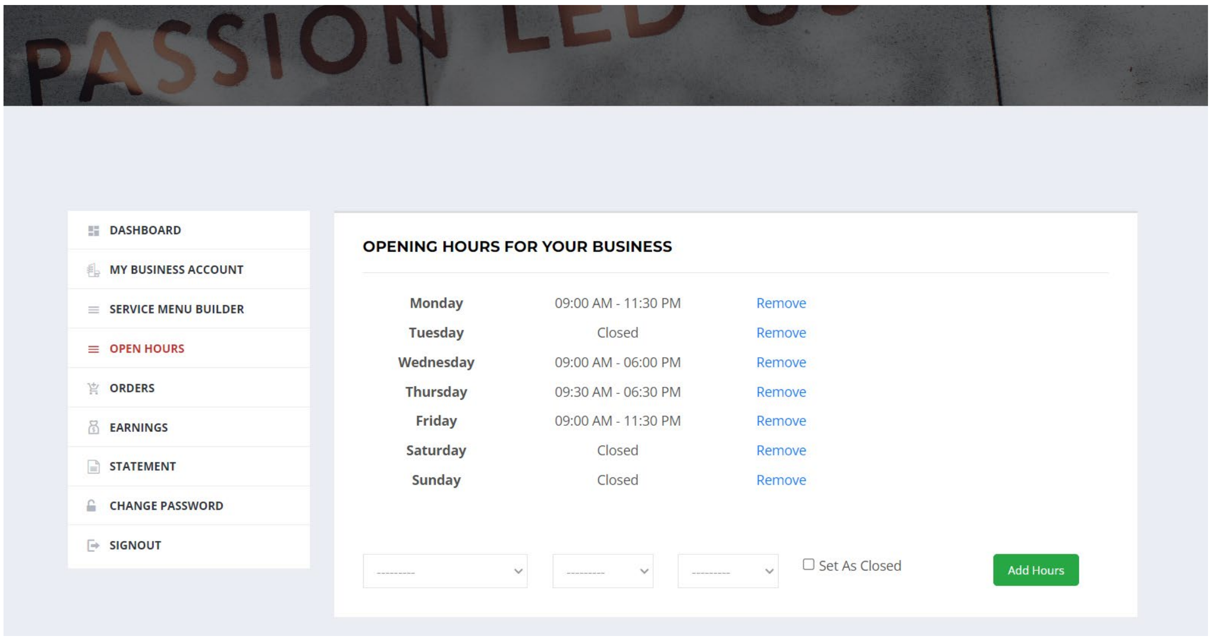
city*
Limerick

Post Code*
V94 WD21

Edit business account



business account



OPENING HOURS FOR YOUR BUSINESS

Monday	09:00 AM - 11:30 PM	Remove
Tuesday	Closed	Remove
Wednesday	09:00 AM - 06:00 PM	Remove
Thursday	09:30 AM - 06:30 PM	Remove
Friday	09:00 AM - 11:30 PM	Remove
Saturday	Closed	Remove
Sunday	Closed	Remove

Opening Times

1.0 Conclusions

In conclusion, the Communication Station has demonstrated that it can managed businesses and assist with growing revenue of small medium enterprises, managing ordering, cart functionality and purchase, along with providing a user friendly interface for customers. This project has delivered the objectives as set out in the project brief.

The strengths and advantages of the Communication Station platform is its ability to interact with a number of other platforms and software's applications. This platform can be tailored to extremely specific and marginal business demands, thus providing boutique application for all users. For example from a hairdresser, multinational corporation to a funeral home and farm, this platform can be tailored for special requirements for the business.

The communication station platform is intentionally designed to be accessible to all users regardless of their literary skills, learning abilities or technical savviness.

The platform does require the internet to run it cant be accessed without connectivity which can be a disadvantage for some smaller business that would operate on paper mostly in rural areas.

I would have enjoyed the opportunity to investigate and develop this platform further but due work constraints and family illness, I have not been able to fully realise this exceptional platform.

2.0 Further Development or Research

Using this platform, as a foundation for a greater enterprise the Communication Station could have exponential growth as more business and partners begin using and utilising the platform. At the moment, I have two small businesses that are interested in using this platform for their online business platform and management tool. In time to come, this has the potential to grow into a large business create employment, revenue and overall expansion of the online system.

As the communication station system becomes recognised and integrated in other industries and platforms the functionality of the system will expand to meet the demand of the required industries.

3.0 References

https://www.toptal.com/developers/gitignore/api/django

https://www.sourcecodester.com/python/14542/food-ordering-system-using-python-source-code.html

https://docs.djangoproject.com/en/4.2/topics/http/middleware/

https://www.sciencedirect.com/science/article/pii/S0360835223002048?

https://codeinstitute.net/ie/blog/what-is-vs-code/#:~:text=VS%20Code%20can%20be%20installed,Java%2C%20Python%20and%20many%20more.

Reference: https://wpengine.com/resources/why-you-should-start-using-font-awesome/#:~:text=Designers%20love%20the%20use%20of,quality%20iconography%20on%20every%20device

https://docs.djangoproject.com/en/4.2/ref/databases/#:~:text=PostgreSQL%20notes,3.1.8%2B%20is%20recommended.&text=Support%20for%20psycopg2%20is%20likely,some%20point%20in%20the%20future.

Installation - https://www.youtube.com/watch?v=qeSzBXsjVzY

https://www.infoworld.com/article/3239675/virtualenv-and-venv-python-virtual-environments-explained.html vir env

python découple 0 https://pypi.org/project/python-decouple/

Create dashboard - https://medium.com/@appseed.us/django-dashboards-open-source-and-free-projects-1d8e64919e6d#:~:text=%E2%9C%A8%20Django%20Volt%20Bootstrap%205,date%20pickers%2C%20and%20so%20on.

Dataset for assisting in creating the dashboard - https://www.kaggle.com/datasets/ahsan81/food-ordering-and-delivery-app-dataset
Use signals in Django - https://docs.djangoproject.com/en/4.2/topics/signals/ for use updates

https://docs.djangoproject.com/en/4.2/ref/forms/validation/ form validation messages – used forms.py and registerUser.html
creating messages - https://ordinarycoders.com/blog/article/django-messages-framework

create a new logo - https://express.adobe.com/sp/design/post/urn:aaid:sc:EU:b171206f-97af-4c79-92c1-f66c938660bb/?workflow=logomaker&autoDownload=true

https://www.rockandnull.com/django-email-verification/ create email verification

could not get the user profiles values to pass from the models.py to the cover.html for the dashboard, passed values using the context processer: https://djangocentral.com/how-to-create-custom-context-processors-in-django/
GDAL: https://gdal.org/index.html
– API Key for Google Maps API
---Google Autocomplete Video---

https://www.youtube.com/watch?v=c3MjU9E9buQ

https://stackoverflow.com/questions/46988817/can-i-limit-google-maps-api-autocomplete-results-to-ireland-and-northern-ireland

https://developers.google.com/maps/documentation/geocoding

---Script to load in HTML head---

```
<script
src="https://maps.googleapis.com/maps/api/js?key={{GOOGLE_API_KEY}}&libraries
=places&callback=initAutoComplete" async defer></script>
```

Sweet alert:
https://sweetalert2.github.io/

error with GDAL: https://gis.stackexchange.com/questions/407782/whl-is-not-supported-wheel-on-this-platform-when-pip-installing-gdal-on-windows

https://stackoverflow.com/questions/18078871/hide-check-radio-button-with-csscss

date time to string for order and checkouts: https://www.programiz.com/python-programming/datetime/strftime

simple json library https://pypi.org/project/simplejson/ The encoder can be specialized to provide serialization in any kind of situation, without any special support by the objects to be serialized (somewhat like pickle). This is best done with the `default` kwarg to dumps.

The decoder can handle incoming JSON strings of any specified encoding (UTF-8 by default). It can also be specialized to post-process JSON objects with the `object_hook` or `object_pairs_hook` kwargs. This is particularly useful for implementing protocols such as JSON-RPC that have a richer type system than JSON itself.

Create PayPal buttons: https://www.paypal.com/buttons/

Code taken from - https://developer.paypal.com/demo/checkout/#/pattern/server
Create PayPal sandbox account: https://www.youtube.com/watch?v=yqTWcayeIdI

Cross site protesting django : https://docs.djangoproject.com/en/4.2/ref/csrf/
Security feature

Paypal sandbox : https://www.sandbox.paypal.com/ie/home

https://www.pinterest.ie/pin/760756562040705479/ pinterest for the poster

4.0 Appendices

4.1. Project Proposal

National College of Ireland

Project Proposal

Business Management Application

17th of December 2022

Computing Part Time

Software Development

Year 2022/2023

Kerrie Newman Ryan

X19135645

X19135645@student@ncirl.ie

Contents

5.0 Objectives

With all businesses, the key concern or the key element to businesses running smoothly is communication. Communication between departments is necessary, but it often falls short between related departments in the same business. The main objective of this project, is to create a Business Application that improves communication for a diverse range of industries, for examples gyms, schools, retail, hospitality even universities. This project will deliver this software using the hospitality industry as its worked example, however, this software will be tailored to any other company, institution or industry as required.

This computing project aims to bridge the communication barrier using technology between departments. For example, with a restaurant chain, there are many different departments or sections that are related in some way but would not have a direct line of communication to each other; kitchen, front of house, marketing, purchasing and Human Resources. This software would make it much easier for each of these to update and correspond internally. So with stock control, purchasing and marketing would communicate with the kitchen and the front of house team would know what menu items were available and what were not available for the restaurant reservations. Essentially building an automated communication between departments, but removing redundant connections. When an object is updated in one department only the related objects in the other departments are updated. The main goals of this as below:

- Automated updates between the related department objects
- User friendly
- Sustainable Development
- Easy set up and training on the user of the software

This application will be deployed on a Raspberry Pi console where fingerprint sensors will be used for authentication for each end user.

6.0 Background

Working for HubSpot, I saw an opportunity to create an application that could be implemented within the Customer Relationship Management (CRM) tool. HubSpot is primarily use for inbound marketing with their customers, however it can also be used for managing marketing and lead management for schools etc. This would work with other integrated software and work for any business. With HubSpot, as its specific with marketing there is an gap in the permissions between departments and teams where related data is only available to the related teams within the CRM. This means that with reporting or ordering or if cross collaboration is required then the information needs to be requested from the related team. With this, the idea came to create a business application where objects related to different teams or different departments within the businesses could be updated cross functionality without the need to request the data, this is all automated from the main object, main input or from the database. It can be created locally over different devices and machines and include the option of this operating over a cloud application.

Meeting the objectives of this project, will require using project management tools like Monday.com to create a Gant chart and working on timelines to create clear concise goals and targets ensuring each object is met. First step of the project is to learn the programming languages which will be used in the project and then working on the database. Next will be creating the front end design for the application and then creating the relationship of data between each department, one to one, one to many and many to many data relationships.

7.0 State of the Art

This application is very similar to CRM technologies like Salesforce, HubSpot and Asana but they depend on cloud and internet technology and can also have a high cost to run and manage. This computing project is set out to bridge the gaps in the CRM and assist with creating the software where the backend can be implemented on a hard ware device that can be managed and maintained locally. While also giving the option to run the application over a cloud connection. During the development and testing phases a raspberry pi will be used for the local machine and server connection.

The main difference from these other technologies and the computing project is giving the user the ability to have the software apply to any operations plan for the business. This stands out for its scalability and sustainability, while also making it user friendly for users using the application and creating internal processes for their day to day running requirements. The main goal is having a single point of reference for users across the application. So been able to use the application in the restaurant industry with different sections from a single point for example a point-of-sale system (PoS) and it updates across other sections from inventory and kitchen management. Removing redundant technologies like excel for record keeping and data entry keeping all data in the same database. The business application or computing project will be reliable for reporting, business analytics; while also assisting with the management of inventory, people and the overall business.

8.0 Technical Approach

Throughout the development process the Agile method will be adapted throughout the entire project process. Working with my project supervisor, the process will involve taking several iterations or incremental steps towards the completion of the computing project/business application. Taking a iterative approach and adapting this method to the project management allows for testing and flexibility throughout the computing project. Using Trello and Monday.com to assist with managing the project creating user stories using agile methods with Kanban boards and Gant charts.

Defining the requirements of the project, overall the computing project will be a transferable software that can be used across many different businesses, but for the overall development and testing the focus will be for one business hospitality (hotel and restaurant management). Throughout the project each objective and goal will use the SMART method, Specific, Measurable, Acceptable, Realistic and Time Based.

During the testing phase there will be a testing for labelling and updating the software for other examples businesses/industries Gym, Outdoor store and Supermarket. Each business has it day to day running operations which would normally include the below:

- Point of Sales, for each front of house department (Hotel and Restaurant)
- Front of House Management (table bookings, reservations)
- Kitchen Management (Inventory, Ordering)
- Hotel Management (Room bookings, Cleaning)
- Staff Management (HR, Rosters, Payroll)
- Multi Software integration – adding a CRM system
- Main Business Database
- Back Up Database for the entire system

The priorities for this business application would be to create the relationships between the point of sales, the front of house tool and the main database. Prioritising learning the different programming languages in order to create the main database for the computing project/business application then designing the relationship from the application to the database. Using the Sprint Agile method and planning the project roadmap features within Jira and Trello each task and goal will be broken into research, development, testing and documentation. Using a burn down report to maintain timing and deadlines.

Breaking it down to the main achievable goals for the project below:

- Creating the fundamental features of the application – CRUD (Create, Read, Update, Delete)
- Creating a back end database for the CRUD application (MongoDB and MySQL)
- Create a User Interface
- Add the application on a Raspberry Pi console
- Update the relationships between the business departments
- As each phase is completed test the application before the next step.

9.0 Technical Details

The primary implementation language for this project will be Python as it has many packages available, its dynamic and has serverless frameworks available which will assist with creating the application in server independent environment. It is the most user friendly language and best for creating software applications machine learning modules. Also, with the application one of the main features would be reporting and analytics Python is ideal for data analytics as well. Overall, Python is the most sustainable and robust option for development, some of the packages and functions that will be used throughout development are listed below:

- Node.js
- Built-in Exceptions
- Numeric and Calculation Properties
- Data Types
- Functional Programming Modules
- File and Directory Access
- Text Processing Services
- Binary Data Services

Data Structures that will be used throughout development:

- Dictionaries
- Maps, hash tables, lists,
- Stacks, Queues
- Tuples and Sequences
- Sets and Looping

The primary Python Libraries that will be used throughout development are Pandas for the data distribution, simple data modelling, data manipulation and cleaning to assist with reporting. Numpy for quick built in computational functions it takes up less memory and is more convenient and faster to use. Scikit Learn for learning algorithms and machine learning to assist with making suggestions for order and inventory. With this Eli5 will also be used as it works in conjunction with the Scikit Learn package and aids with debugging the machine learning models and algorithms. Requests as it sends and creates HTTP requests to the python applications creating communication and assisting with API requests. PeeWee for object mapping and database support for creating and maintain the databases. Pillow for editing and adding images to the GUI and Tkinter for creating the GUI for the application. Pytest for testing the application. With this Django will also be implemented through the application, for user authentication, security, scalability and the database can be implemented with MySQL and MongoDB.

As the business application will have a variety of functions, the primary algorithms been used throughout development are:

- Sort Algorithm – Merge, Quick, Heap, Count Sort
- Search Algorithm – Binary and Hashing
- Knuth-Morris-Pratt Algorithm
- Language Detection Algorithm
- Dijkstra

10.0 Special Resources Required

The special resources required and requested is a Raspberry Pi 4 with hand and fingerprint sensors. The main innovative step of the project is to implement the full operational application on a raspberry pi with an operating point of sale system. The sensors within the project will be used as authentication for the end users from each department user.

For testing and deployment, Heroku and Amazon Web Services will used, due to the scale of the development. The extra add on will need to be purchased in order to avail of the full resources and services.

11.0 Project Plan

The project timeline is beginning development in December with 5 phases with testing throughout each phase ensure the minimum bugs.

Phase One: Database creation

Given that the application will have many database features and relations for each business and different departments within the application, two options will be used for the database NoSQL(MongoDB) and MySQL. MongoDB due to its flexibility (Insane Schema) when datasets and data will constantly changing this would be more durable for these changes. Whereas the relational model will have the table views, indexes and data relationships between these table views, which makes it easier managing the logical structure of the data storage. So that each department can store their related data in the database. Working with open source data in https://data.world/datasets/restaurants and creating tables and then the relations. Testing that that database is functioning after this phase is complete.

Phase Two: Create the basic application processes (Create, Update, Read, Delete)

With Python and Node.js begin creating the CRUD application of the complete application and associate the application to the database. Once the application is connected and functioning, testing the connection and durability of the application and database connection, ensuring that there are no bugs and that the data is appearing correctly.

From here duplicating the CRUD so that each department has its own application and relationship to the database. Testing the implementation of the application and function of the department to the database.

Once functioning begin the different roles within each department:

- Super user – user with full access to all of the application and database (this would be open for owner of area manager of the business for example that would have full access and scope of the application and have the ability to edit the department and users permissions and relationships)
- Data administrator - (database permissions giving ownership to read, update, edit and delete the database)

- Manager or Department manager - (person that over sees the department can edit, read and delete information on the application)
- End user - (staff member, bar person, receptionist etc permissions to read and use the app for bookings or as a point of sale)

In the next phase the login authentication will be developed through the raspberry pi console sensors.

Phase Three: Raspberry Pi console development

Creating the raspberry pi console itself, so building and adding the sensors ensuring the sensors work with a test fingerprints before implementing the application on the raspberry pi.

Phase Four: Implementation of the Business Application on the Raspberry Pi Console

Implementing the test application on the raspberry pi console. The innovative part of having fingerprint sensors so that each user is authenticated for their specific department with their specific permissions.

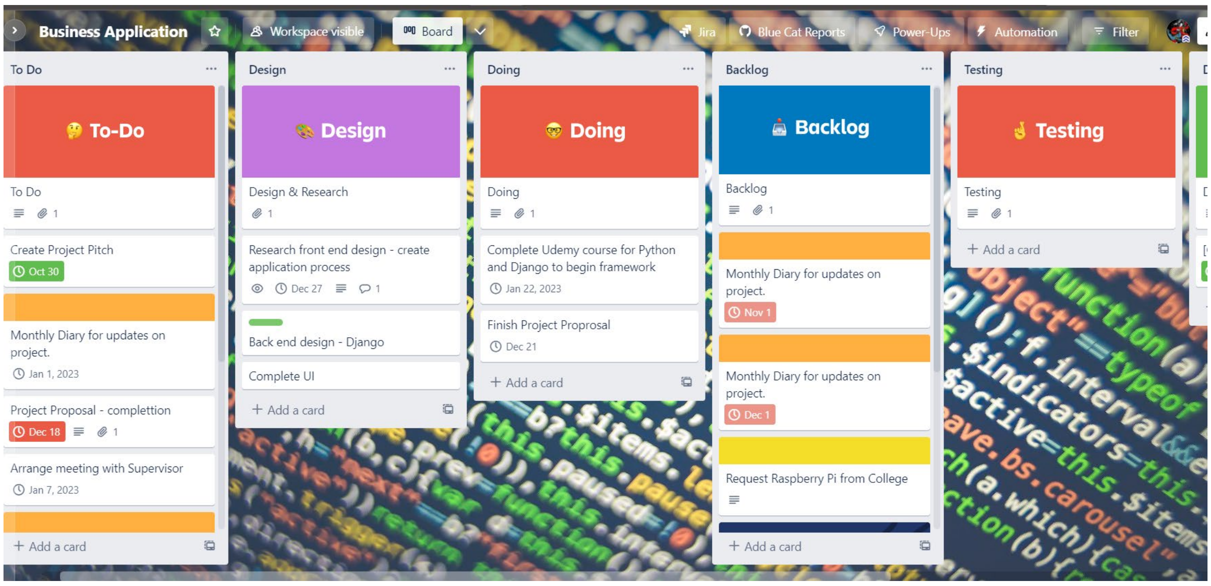
As the initial setup with have the super admin permissions the next step would be adding the end user with read and update access only. Then moving on to each new user on the raspberry pi console.

Phase Five: Full application deployment (with testing and debugging)

This will be the final stage of implementation on the raspberry pi console with full deployment of the application. With this, testing and debugging of the complete application on the raspberry pi device, ensuring all of the objective have been met and that the application is fully operational from each users standpoint.

The final phase of the project is creating the documentation for the Business Application.

Below pictures showing Gant chart and Kanban board of project timeline and milestones:


Business Application | Workspace visible | Board | Jira | Blue Cat Reports | Power-Ups | Automation | Filter

To Do
- To-Do
- To Do — 1
- Create Project Pitch — Oct 30
- Monthly Diary for updates on project. — Jan 1, 2023
- Project Proposal - copmlettion — Dec 18 — 1
- Arrange meeting with Supervisor — Jan 7, 2023
- + Add a card

Design
- Design
- Design & Research — 1
- Research front end design - create application process — Dec 27 — 1
- Back end design - Django
- Complete UI
- + Add a card

Doing
- Doing
- Doing — 1
- Complete Udemy course for Python and Django to begin framework — Jan 22, 2023
- Finish Project Proprosal — Dec 21
- + Add a card

Backlog
- Backlog
- Backlog — 1
- Monthly Diary for updates on project. — Nov 1
- Monthly Diary for updates on project. — Dec 1
- Request Raspberry Pi from College
- + Add a card

Testing
- Testing
- Testing — 1
- + Add a card

Fig. 1 Image from Trello Planning Board

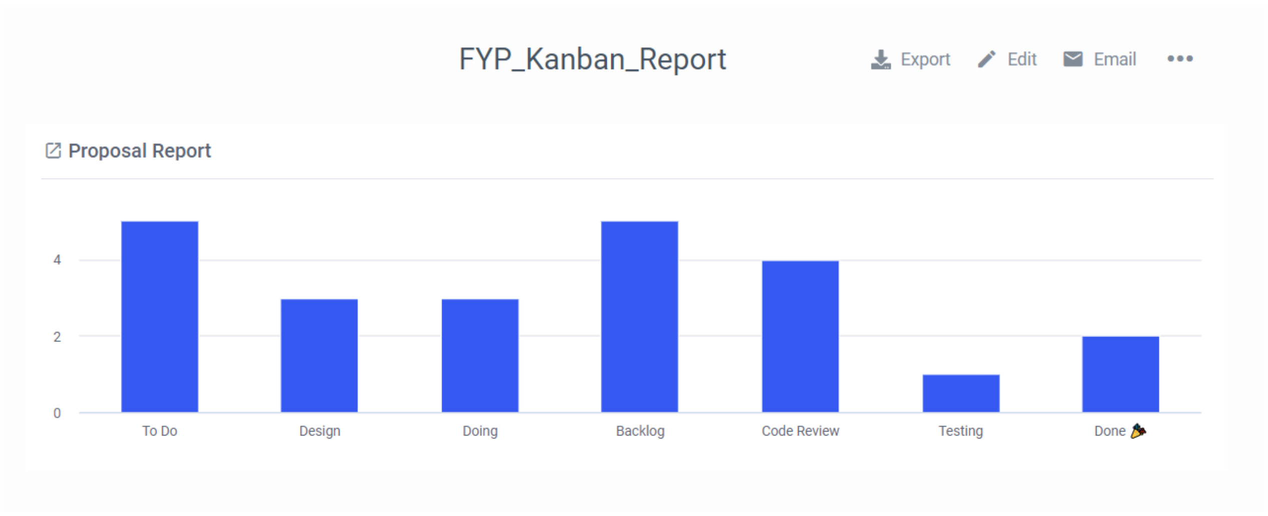


Fig.2 Kanban Report from Trello showing number of tasks on each card with the board

Business_App_Gant_C
hart-2023_Completior

Attachment of Excel File showing Gant Chart for Business Application Completion

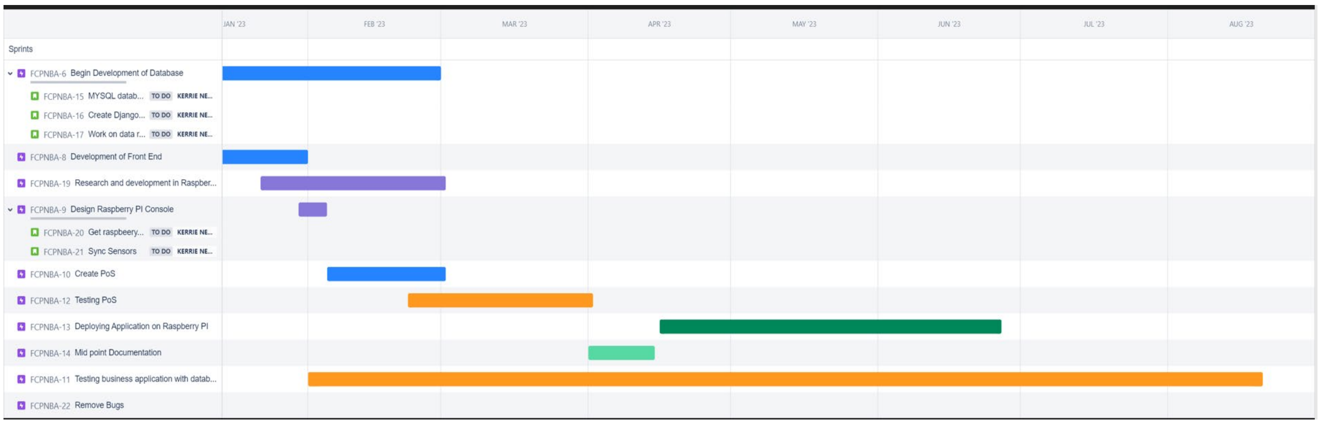


Fig.3 Sprint Chart and Gant Timeline

12.0 Testing

This project is using the agile method with this the TDD method (Test Driven Development) meaning that after each phase is completed the system testing types will be used. With each end user the permissions will be tested along with the allowed features for each user. For testing user profiles will be created for each use case mentioned in section 7 "Project Plan".

The stage will consist of "Coding, Create Test, Test Fails, Refractor the code or Test Past". Creating test cases and recording each test result in a report under the headings "Test Case", "Test Coverage or Actions", "Execution Details and Results".

Once the each phase of the stage has been completed the below testing phases will be used:

- **Functionality Testing** – ensure that the phase of the development that the application and goal is functioning and is completing the needed tasks
- **Performance Testing** – that the application is performing as needed before beginning the next phase of development.
- **Sanity Checks** – when the unit build is completed and a change has been made with the code this checks the modifications did not impact the function of the application
- **Scalability Testing** – ensure that its suitable under user and resource scaling before moving onto the phase of development

When the application has been completed and fully implemented in the raspberry console the complete application will be tested. The application will be tested for bugs the users permissions in relation to each department part of the application.

- **Graphical User Interface(GUI) Testing** – that the front end works and carries out the actions that each button and user inputs to the application
- **Testing for Compatibility – that the application can be used and complete the required actions correctly in the app**
- **Handling Exceptions** – if the user inputs the wrong action or an error occurs on the platform that the proper error message appears.
- **Stress Testing – that the application can handle having more than one user using the system at the same time**

Thorough testing using the agile method (TDD) will ensure a successful project outcome.

12.1. Development Screenshots

Screenshots_Dev_4th – URL to folder with screenshots from development

13.0 Reflective Journals

Supervision & Reflection Template

Student Name	Kerrie Newman Ryan
Student Number	X19135645
Course	BSHCSDE4
Supervisor	Enda Stafford

Month: October, November 2022

What?

Reflect on what has happened in your project this month?

For this month the main focus was coming up with an idea for the 4th year project. Over the summer break the initial choice for my 4th year specialisation was Machine Learning and Data Analysis. The initial project was an analysis on sports injuries and the initial sport, so for example combat sports and rugby. With the change in the specialisation to Software Development the idea was changed and the project pitch was focused on.

Some of the ideas that were explored were:

- Gym App for specific sports to help with preventing injuries (it was too general)
- Music App for learning instruments (not enough data to help with learning instruments or specific instruments)
- A learning to drive application (too general as well)
- Beer brewing application (too general and would need a lot of IOT technology that I don't have I have access too)

In the end we decided that creating a UI system for businesses similar to a CRM system but different in the sense of multi department communication and storage data. It would work from permissions between staff and customer and what data they can and cant access. This would help with GDPR and other ethics and governance in businesses.

The focus this month was approval for the project and creating the project. The main focus been what technologies we can use to achieve this and the extra training and research needed to complete this project

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

Successes

Getting project approved and figuring out the technology that is required and used for the system. For the best back end framework in this case Django and PostgreSQL would be great for development. So beginning the tutorials to beginning learning how to apply these frameworks: https://www.w3schools.com/django/django_intro.php

Starting the documentation and research for the project

Challenges beginning the development with the additional coursework and meeting the deadlines.

Now What?

What can you do to address outstanding challenges?

Working on time management and creating a time table to keep up with work full time, the module requirements and projects and the computing final year project. Keeping active during development so when any issues arise it can be dealt with easily and quickly.

Student Signature	Kerrie Newman Ryan

Supervision & Reflection Template

Student Name	Kerrie Newman Ryan
Student Number	X19135645
Course	BSHCSDE4
Supervisor	Enda Stafford

Month: December/ January 2022

What?

Reflect on what has happened in your project this month?

- Assigned and met with Supervisor
- Selected topic and was approved for software system

Creating the planning and project proposal for the due date on the 21st of December. The project itself has been fully approved.

Since the idea is now clear and the planning is taking place, its easy to lose sight of the core goal for the computing project. At the moment I am completing the Django boot camp so as to upskill in the technologies to use in the course.

- https://www.udemy.com/course/python-and-django-full-stack-web-developer-bootcamp
- https://www.w3schools.com/django/django_intro.php

Planning to the development during the year is been worked on and taking courses to make a clear understanding of goals and what can be achieved within the project. The month was also spent working with my supervisor on how to best approach creating the database, are the tools I am using going to showcase the best innovative project. Hoping to use some of the learned algorithms and best practices so as to create the best outcome for the project and meeting the project proposal requirements.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

Successes

Having a clear outline for development. Using project management tools Trello, Jira and Monday.com the timeline for the project development and testing has been created. This can be adhered to directly and help with any changes in time table, work or outside factors during the creation of the project. A Gant chart has been created as well in different formats to help with the timeline of the development and testing.

Getting the project proposal done and submitted by the due date, getting the documentation ready and studying applicable use cases that would be related to the project.

Challenges

Meeting all for the deadlines for each of the modules and also ensuring work commitments makes the time management difficult for the project itself. At the moment its working towards deadlines for parts of each module and then setting the time aside for the project. This will be a challenge going forward throughout all of the college year.

Now What?

What can you do to address outstanding challenges?

Using the external tools and adhering to the timetable and gant charts created, and adapting an agile method throughout development. Keeping to the plan created and deterring on a different path and creating a different project will help immensely going forward with the creation of this.	
Student Signature	Kerrie Newman Ryan

13.1. Other materials used

Any other reference material used in the project for example evaluation surveys etc.