# ComeChop Ecommerce Site (CES)

Anthony Aneke

Full Stack Diploma

In Software in Development

2020/2021

Software Architecture Document
and Use Case Specification

Date: April 21, 2021

**Acknowledgements**

I would like to express my sincere gratitude to my mentor, Caleb, Code Institute Learning respective lectures: Brian O'Grady and co-workers at work, for their continuous support of my Post graduate Diploma - particularly the Django full stack milestone project, and their patience, motivation, and immense knowledge. Their experience, time and guidance has helped me throughout my research and execution of this project.

## Document source

This document is maintained in GIT under the Django milestone project folder.

**Revision history**

| Version number | Date | Summary of changes | Reviewer Initials |
|---|---|---|---|
| 1.0 | 02/02/2021 | Initial Version | AA |
| 1.1 | 02/06/2021 | Revision in Elaboration | AA |
| 1.2 | 02/11/2021 | Incorporated internal review changes | AA |
| 1.3 | 02/16/2021 | Updated formatting | AA |
| 1.4 | 02/23/2021 | Modified for Data view, interface design changes | AA |
| 1.5 | 03/2/2021 | Minor updates | AA |
| 1.6 | 03/7/2021 | Updated the Login Use Case document name in the reference section | AA |
| 1.7 | 03/2/2021 | Revision to add content for Data View and interfaces | AA |
| 1.8 | 03/2/2021 | Reviewed and Updated Section 4 Use Case View<br>Updated Section 5.3<br>Updated Section 6 (Process View) | AA |
| 1.9 | 04/30/21 | Waited for feedback from Datacentric milestone project and finally received it and incorporated feedback into Django full-stack milestone project | AA |

**Distribution**

This document has been distributed to:

| Name | Function |
|---|---|
| Brian O'Grady | Programme Director, Code Institute |
| Nakita McCool | Technical Program Manager, Code Institute |
| Mark Rudge | Senior Product Developer, Code Institute |
| Ulysses Ryan- Flynn | Student Care Adviser, Code Institute |
| David Howard | Education Adviser, Code Institute |
| Caleb | Mentor, Code Institute |

# Table of Contents

# Software Architecture Document

## 1. Introduction

The ComeChop Ecommerce Site (CES) Software Architecture Document (SAD) provides a comprehensive architectural overview of the CES, using a number of different architectural views to depict different aspects of the CES. It takes the use cases as the guiding source to come up with an CES architecture that satisfies the business and final project requirements.

### 1.1 Purpose

This document provides a comprehensive architectural overview of the CES. It is intended to capture and convey the significant architectural decisions which have been made on the CES.

This document discusses the software components and their position in the overall architecture of the CES. This document describes the logical view, physical deployment view and the data view of the CES.

This document is intended for a technical audience who want to have an insight into the logical and physical aspects of the CES and have a broad understanding about the proposed solution in terms of data model, reference architecture of the static website template (see reference section), critical use-case flows and architecturally significant components of the software.

The audience for this document is the technical lead, developers, database designer - who are the key players in developing and implementing the CES solution. This document also helps the infrastructure team understand the infrastructure for the various deployment regions.

### 1.2 Definitions, Acronyms, and Abbreviations

| | |
|---|---|
| API | Application Program Interface |
| AWS | Amazon Web Service |
| CES | ComeChop Ecommerce Site |
| DBaaS | Database as a Service |
| JAWSDB | JAWS Database |
| MySQL | My Structured Query language |
| ORM | Object-Relational Mapper |
| PaaS | Platform as a Service |
| RDS | Relational Database Service |
| RDBMS | Relational Database Management System |
| REST APIs | Representational State Transfer Application Program Interface |
| SQL | Structured Query language |

### 1.3 References

The following documents were used as a basis for this document:

**Table 1-1: Reference**

| Ref. ID | Reference Title | Repository | File Name or URL |
|---|---|---|---|
| Ref 1 | Code Institute Lectures | Code Institute Learning Management System | https://learn.codeinstitute.net/login?next=/ |
| Ref 2 | Use case specification | GIT | |

| Ref. ID | Reference Title | Repository | File Name or URL |
|---|---|---|---|
| Ref 3 | Design Patterns | SAD and GIT | Brian O'Grady lecture on Model View Controller from the Learning Management System, Code Institute<br><br>Introducing System Development, by Steve Skidmore and Malcolm Eva, Palgrave Macmillan<br><br>UML2 and the United Process: Practical Object-Oriented Analysis and Design, Jim Arlow and IIa Neustadt, Addison-Wesley<br><br>Patterns of Enterprise Application Architecture (The Addison-Wesley Signature Series), Martin Fowler, Addison-Wesley. |
| Ref 4 | CES Deployment Plan | SAD and GIT | |
| Ref 5 | CES Data Model Document | SAD | |
| Ref 6 | Interface Model | SAD | |
| Ref 7 | Prebuilt Checkout page | | https://stripe.com/docs/checkout/integration-builder |

### 1.4 Overview

The rest of the document describes the CES architecture from different viewpoints, the architectural goals, constraints and the tradeoffs, if any, in selecting a particular architecture. The document is organized in the following sections:

- Architectural Representation: Describes the different views used to portray the CES software architecture.
- Architectural Goals and Constraints: Identifies the key architectural goals and constraints that drive the design and development of the CES solution.
- Architectural Trade offs: Describes the tradeoffs, if-any.

## 2. Architectural Representation

The CES software architecture provides a holistic view of the software components and the interactions among them. The software architecture is represented by five "views" that provides an insight into the system architecture from various perspectives: use-case, logical, process, deployment, implementation, and data.

Each view of the software architecture is designed to provide perspective and understanding on a different aspect of the CES. The individual views give the perspective from a specialized point of view but the true value of the views is realized when they are combined and presented holistically. This integration further reassures that the user needs are realized, and it also helps to drive the technical tasks that need to be performed to design and develop the CES.

The CES software architecture is represented in the following views in this document:

- Use-Case View: This view represents the business processes from the use-case perspective that is critical to the CES.

- Logical View: This view shows some high-level software component hierarchy and describes some architecturally significant ones.
- Deployment View: This view provides the deployment model for the CES in the Development, QA and Production environments, and describes where the various software components reside from a technical standpoint.
- Implementation View: This view describes the CES in terms of various layers, i.e., user-interface, application layer, data layer and such.
- Data View: This view represents the CES data model where the physical data will reside.

## 3. Use-Case View

CES has two high level use-cases that map to the three phases of the CES shopping online process. These two are then further decomposed into sub use-cases. Detailed use-cases can be found in the Manage recipe online shopping and Make Purchase use case specifications document.

**Manage Recipe Online Shopping**: This use case describes how a Web Customer uses the ComeChop ecommerce web site to make a recipe purchase online.

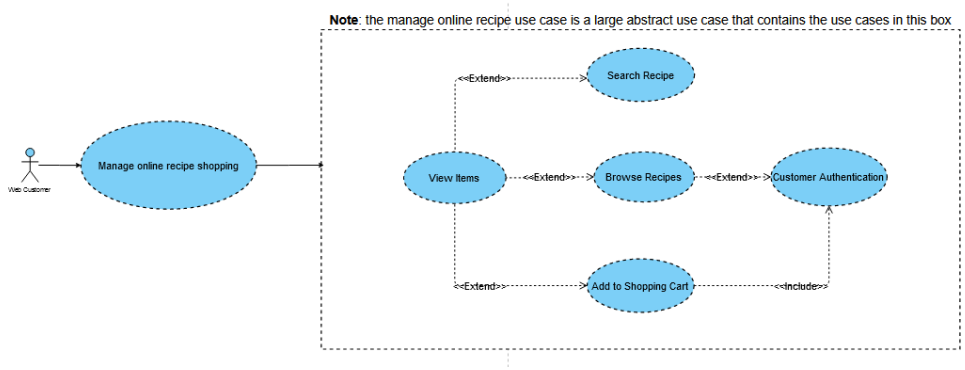


**Figure 1 - Manage Recipe Online Shopping**

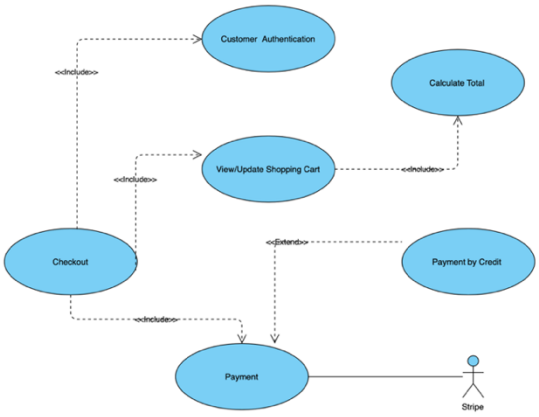


**Figure 2 - Make Purchase Use Case**

# 4. Logical View

## 4.1 Overview

This section provides the logical view of the CES site. Since this application is based on a framework, it helps realize the use-cases and the individual processes.

## 4.2 Architecturally Significant Design Components

The significant design components for CES are broken down according to the various stages:

Database – This is the database integration to a relational database.

Payment processing system – The application integrates via the Stripe Application Program Interface (API)

# 5. Process View

Detailed process flows are described in the CES Use-Case Specification section below.

# 6. Deployment View

The figure below provides a pictorial view of the software deployment of CES in various build stages.

Figure1-3: Deployment View for CES

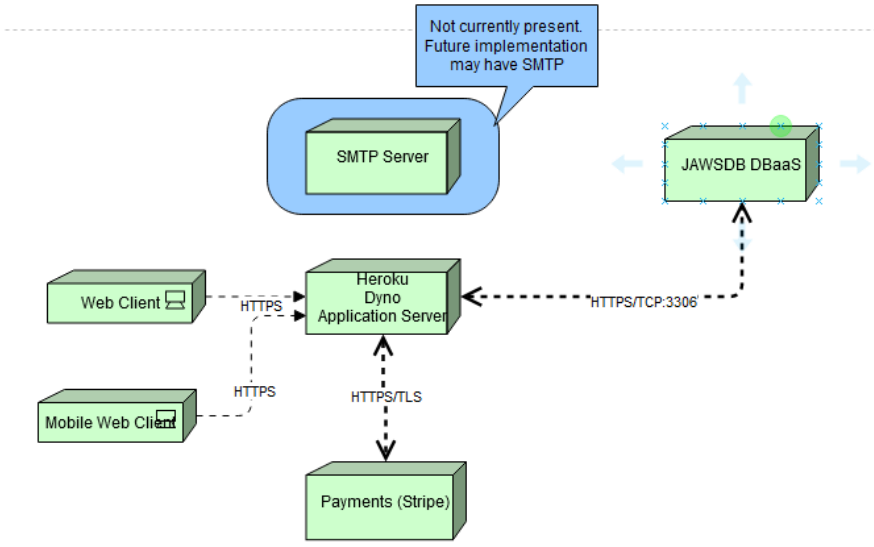


**Figure 3 - Deployment View**

The main components of the CES Full Stack application are mentioned below:

- Workstations, Laptops and Hand-held devices: These are used to access the CES application running on the application server. The application server used was Heroku platform as a service.
- Heroku Application Server: This server hosts the CES application and provides components that connects to MYSQL (See below for more details).
- Database Server: MYSQL listens to a TCP port on the operating system. This port number is 3306.

7. Implementation View

The CES will follow a three-tiered architecture composed of user interface layer, business layer and database layer. Figure 8.1 shows a graphical representation of the CES layered architecture.

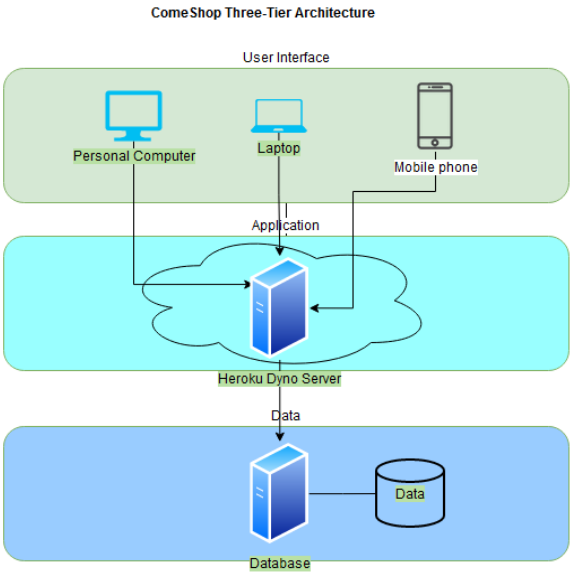


**Figure 4 - CES three-tier architecture**

## 7.1 Overview

The CES will follow a three-tier architecture composed of a User Interface Layer, Application Layer and the Data Layer. The following sections describe the various layers and the interaction between them. The layers interact with each other to let an CES user upload and online shopping related tasks. Figure 8-1 shows a high-level view of the components in the various layers and how they interact with each other.

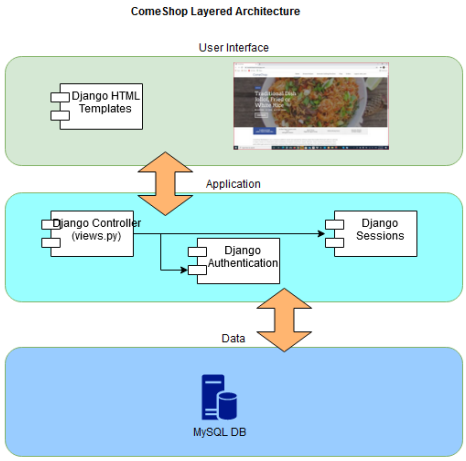


**Figure 5 - CES Layered Structure**

## 7.2 Layers

**User Interface Layer**: This layer deals with the look and feel of the application and is the medium through which users interact with the application. The CES is accessible through a standard web client. The main components for this layer are the UI components (e.g. HTML, CSS, JavaScript, images etc). Any actions on this layer invoke the application layer where the business logic is coded and located.

**Application Layer**: The Application Layer contains the business logic and the process flows for the acquisition of recipe and upload of recipe. The Application Layer links with the data layer to store and retrieve data. The data is stored into persistent storage (i.e. database). The main components in this layer are the controller classes for the application layer and for application logic, these components perform the actual actions initiated by the web customer actor through the UI layer. This layer also utilizes the business processes and the various cloud components (e.g. Python+Django, Heroku (PaaS), Stripe Payment API etc).

**Data Layer**: This layer stores the uploaded recipe data in the physical database. The Data Layer servers as a repository for the procurement of recipes. The Data Layer has data access objects performing the read/write to the physical storage. The main components of the data access layer is provided by the Django object relational model (ORM).

### 7.3 Design Patterns

The CES application uses the MVC Design Patterns.

#### 7.3.1 MVC Design Pattern

The Model-View-Controller (MVC) pattern is the most common of the patterns that implements the separation of responsibility. This pattern is used to separate data (model) and user interface (view) concerns, so that changes to the user interface do not impact the data handling, and that the data can be reorganized without changing the user interface. The MVC design pattern solves this problem by separating or decoupling the data access and business logic from data presentation and user interaction. This is done by introducing the controller.

**Error! Reference source not found.** provides a graphical representation of the MVC pattern. The arrows indicate a direct association and the dashed arrow indicates an indirect association.

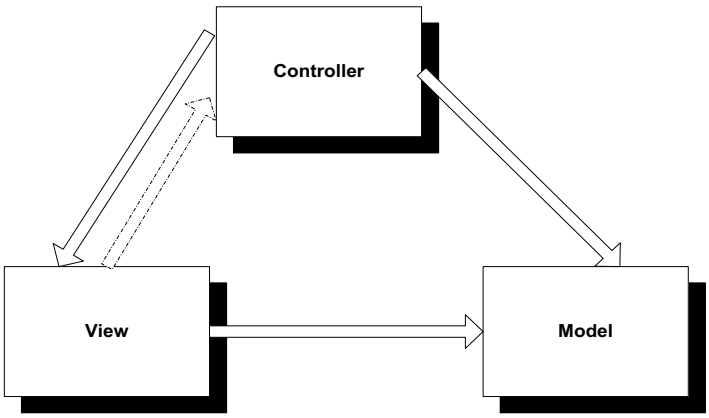


**Figure 6 - MVC Design Pattern (Im**

CES is based on the Django framework which is an implementation of the MVC design pattern.

### 7.4 Interface Design

CES will interface with the JAWSDB Database-as-a-Service MySQL instance for data exchange.
CES utilizes the custom Heroku Django implementation for Django ORM. The Django ORM will interchange messages between the JAWSDB DBaaS and the CES application deployed on a Heroku Dyno.

#### *7.4.1 Receiving Data from JAWSDB*

To retrieve data from JAWSDB, the CES application will invoke the Django ORM interface. The Django ORM interface custom Heroku implementation will in turn retrieve information from the JAWSDB DBaaS.

DBaaS may forward the request to yet another external interface. For the implementation of CES, JAWSDB used an AWS RDS MySQL instance deployed on the US east region. As a result, the JAWSDB will interface with the AWS RDS service via RESTFul API to retrieve transactional data from the MySQL RDBMS database.

### *7.4.2 Sending Data to JAWSDB*

The sending of data to JAWSDB will also utilize the custom implementation of Heroku Django ORM. Django ORM will in turn forward the request to JAWSDB.

Django ORM builds the INSERT, DELETE and UPDATE SQL statements that will be issued through the interfaces to the remote RDBMS.

For this implementation, the DBaaS JAWSDB instance was provisioned as an AWS RDS. Therefore, the JAWSDB DBaaS will issue REST APIs to execute the SQL INSERT DELETE and UPDATE statements on the AWS MySQL instance.

### *7.4.3 Sending and Receiving Data from Stripe Payment*

The interface of Stripe is made available via a REST API. For security, the private key of the credentials is stored in environment variables on the Heroku platform. The public key is embedded in the application and passed to the Stripe interface invocation.

The Stripe interface specification of callbacks allow the Stripe system to return control back to CES. Once the Stripe authentication is setup on the client via JavaScript, control is passed on to Stripe for the collection of payment information and its submission.

Upon success, the Stripe registered callback is invoked returning control back to CES for the display of the payment success and/or confirmation page.

## 7.5 Data View

The MySQL relational database is the system of record for CES data. The data includes recipe information, recipe orders and user credentials. The MySQL database will not hold any credit card or personal information. Although, user's full name and email address are to be stored. The user's passwords stored in the MySQL database are encrypted. CES will obtain the credentials from the database and authenticate users by decrypting the passwords and creating user sessions.

Users of the system will not have direct access to the database. The connectivity credentials are to be established on the Heroku platform via environment variables such that only the application can via a service account access the backend data.

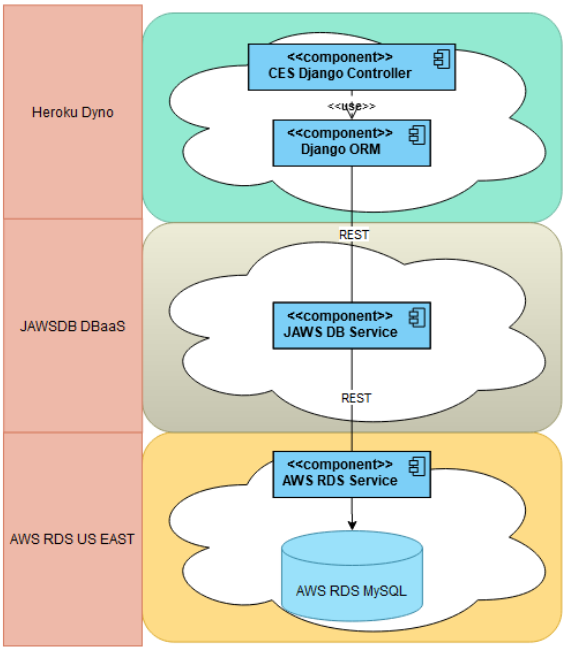


**Figure 7 - CES Data Transfer View**

### 7.6 Data Entity Relationship Diagram (ERD)

The diagram below describes the ERD for the CES

For example:
An instance of the “auth_user” class, is associated with zero to more “recipe” class instances.
Zero or more instances of the “Order_recipe” class, is associated with one “auth_user” class instances.

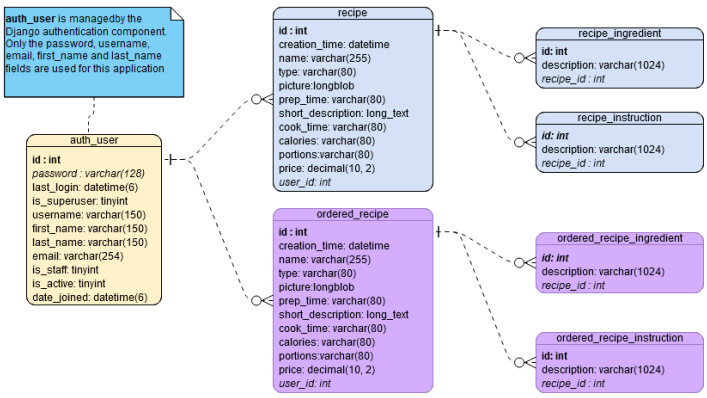


**Figure 8 - Data Entity Relationship Diagram**

# Use Case Specifications

User Stories

As a web customer, I want to:

a) Have the ability to easily find and understand the controls for the application so that I can operate it easily.
b) A large easy to press button and controls, so that regardless of the size of my finger I can use the application.
c) The ability to choose from recipes uploaded by any user, so that I am able to replicate the cooking directions.
d) Visual icons and images that I recognize, so that I understand the recipe and cooking directions.

As a web customer in the capacity of a recipe author, I want to be able to:

a) Visually and operationally interface with an appealing site, so that I have a positive experience when using the site.
b) Create, read, update, delete (CRUD) recipe(s) and generate cooking directions.
c) Search a recipe, but not easy for a child who is randomly pressing buttons to access, so that a profile is not deleted by accident.

## Use Case Specification: Manage Online Recipe Shopping

**8. Use Case Name**

Manage Online Recipe Shopping

**8.1 Brief Description**

This use case describes how a Web Customer uses the ComeChop ecommerce web site to make a recipe purchase.

**8.2 Actors**

Web Customer: Registered Customer and New Customer

**8.3 Preconditions**

- The Web Customer has registered to the site through the Sign-up feature
- The Web Customer has logged into the site
- The Web Customer is authorized to make an online purchase
- The Online shopping ecommerce site must have gathered recipe information from Registered Customers

### 8.4 Flow of Events

#### *8.4.1 Basic Flow*

This basic flow is for the Web Customer to make an online purchase from the Comechop ecommerce website.

*8.4.2 Start of Use Case*

1. The use case starts when the Web Customer clicks on the *Shop* link at the menu bar.
2. The system displays to the Web Customer a list of recipes other Registered Users have uploaded.
3. The Web Customer elects to browse the recipe list shown and elects to click on the indicator (icon) to add to the cart.
4. The Web Customer proceeds to click on the *Checkout* button on the cart presented on the screen.
5. The system displays to the Web Customer the order review page which includes the added recipes deemed for purchase along with the calculated total amount for payment.
6. The Web Customer accepts the order by clicking on the *Checkout* button.
7. The system displays to Web Customer the payment screen.
8. The Web Customer elects to key in payment information i.e. credit card information.
9. The Web Customer provides the name, address, credit card number and clicks on the *Pay* button.
10. The system displays to the Web Customer the payment order confirmation page.

#### *8.4.3 View Ordered Recipe*

1. The Web Customer clicks on the Orders link.
2. The ecommerce site displays ordered recipes made by the logged-in user.
3. The ecommerce site displays a list of ordered recipes for selection.
4. The user clicks on a desired recipe to view the recipe details.
5. The ecommerce site displays the details of the recipe: calories, picture, ingredients, instructions, portions, prep time, cook time, recipe name and recipe description.
6. The Web Customer uses the recipe information to cook the dish.

#### *8.4.4 Edit data*

1. The Web Customer modifies the information to be updated. The system displays the modifications as they occur.
2. The Web Customer clicks on the *Browse Recipes* link.
3. The Recipe site presents a search form with the following fields: recipe category, ingredients, ingredients Boolean operations.
4. The Web Customer selects the search criteria using any of the fields the system provided.
5. The Web Customer elects to click on the “Search for recipe” button.
6. The Recipe site presents a list of recipes that have been previously created by a Recipe user.
7. The Web Customer click on the “Edit” icon on the recipe to edit the Recipe list.
8. The system displays the Edit form.
9. The Web Customer keys in the changes desired on any of the presented fields.
10. The Web Customer clicks on the <Submit Cooking Direction> button.

#### *8.4.5 Save data*

The Web Customer requests to save the changes. The system validates that the values entered are appropriate to the type of data being edited. The system saves the information as pending changes upon successful validation. The system notifies the Web Customer of the successful save.

#### *8.4.6 Use Case Ends*

### 8.5 Alternate Flows

1. At any step, the Web Customer can click out of the *Shop* page and continue browsing the site.
2. At any time, the user may return to the *Shop* page. The recipe site preserves the items previously in the shopping cart
3. At any time in the *Shop* screen, the user may choose to delete items from the cart. Once deleted, the item deleted is removed from the cart. The Web Customer is presented with the remaining items in the cart. When no items are left, the system notifies to the Web Customer that there are no more items stored in the shopping cart.
4. At step 2.1.1:9, if the Web Customer chooses to cancel the payment, the Web Customer is presented with a *Cancel* confirmation page

### 8.6 Alternate Flows

#### *8.6.1 Handle Invalid data*

This alternative flow begins in Save data step of basic flow when the system identifies that invalid data has been entered by the Web customer during the update of recipe information.

8.6.1.1 Identify invalid data

The system encounters data inappropriate to the information being edited and prompts the Web customer to correct the entered data

8.6.1.2 Rejoin the Flow

The use case continues at the beginning of the Edit data step of basic flow.

### *8.6.2 Handle Invalid File Format*

This alternative flow begins in Generate Cooking Direction step of the Basic Flow step when the web customer identifies the need to attach a photo during the Generate Cooking Direction Step.

#### 8.6.2.1 Identify Invalid file

The system encounters an incompatible file format during upload and prompts the Web customer to locate a different file.

#### 8.6.2.2 Rejoin Flow

The system rejoins the alternate flow Generate Cooking Direction Page at Generate Cooking Direction step

### 8.6.3 Quit

This alternative flow begins at any point when the user identifies the need to exit the recipe Information.

#### 8.6.3.1 Complete or Cancel Pending Activities

The Web Customer elects to either complete or cancel the pending activities. The system performs appropriate processing and notifies the Web Customer of success.

## 8.7 Special Requirements

1. Editable Recipe Information:

The system shall provide capability to edit the following:

- recipe label
- recipe type (e.g. lunch, breakfast)
- recipe description
- recipe ingredient
- recipe instruction
- preparation time
- cooking time
- portions
- calories
- price

## 8.8 Post Conditions

- Updated Recipe Information is in the system

## 8.9 Extension Points

- N/A

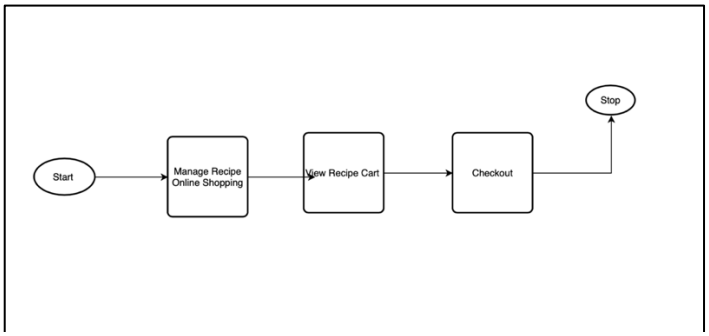


**Figure 9 - Manage Online Shopping Activity Diagram**

## Use Case Diagram

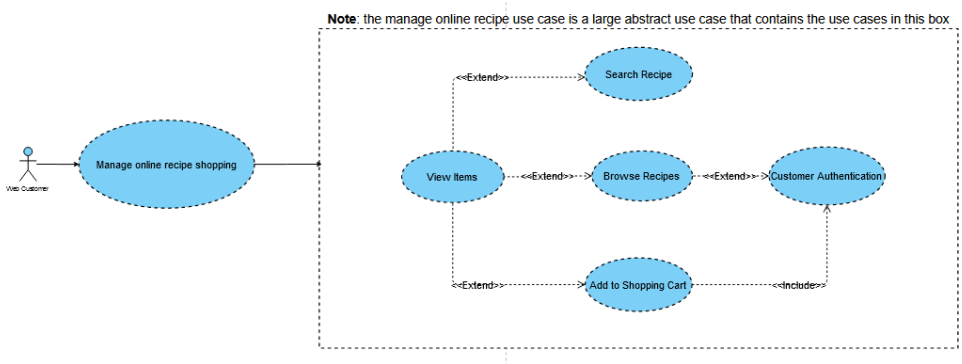


**Figure 10 - Manage Online Recipe Shopping Use Case Diagram**

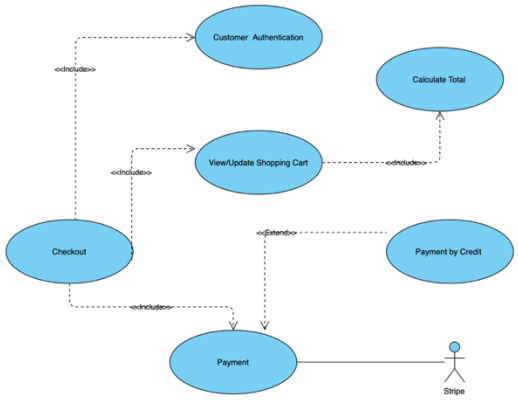


**Figure 11 - Make Purchase Use Case Diagram**

The **Web Customer** actor uses the web site to make purchases online. Top level use cases which are fragments of a larger use case: **View Items**, **Make Purchase** and **Client Register**. The View Items use case could be used by customer as top-level use case if customer only wants to find and see some products. This use case could also be used as a part of Make Purchase use case. The Client Register use case allows customer to register on the web site, for example to upload new recipes or purchase them. Note, that the **Checkout** use case is an included use case not available by itself - checkout is part of making purchase.

# Use Case Specification: Manage Recipe Items

9. Use Case Name

Manage Recipe Items

9.1 Brief Description

This use case describes how a Web Customer manages recipe items on the ComeChop ecommerce web site to make a recipe available for online purchase (refer to Manage Online Recipe Shopping for use case functionality)

9.2 Actors

Web Customer: Registered Customer and New Customer

9.3 Preconditions

- The Web Customer has registered to the site through the Sign-up feature
- The Web Customer has logged into the site
- The Web Customer is authorized to generate recipe

## 10. Flow of Events

### 10.1 Basic Flow

This basic flow is for the Web Customer to manage a common dataset about a particular food domain where the user can generate food directions for their favorite cuisine for anyone who is interested in making that particular dish.

### 10.2 Start of Use Case

The use case starts when the Web customer queries the system to view recipe records. The system displays all recipe records associated with the queried recipe type. The recipe types are specified in the Special Requirements

#### 10.2.1 Generate Cooking Directions

1. The Web customer elects to generate cooking directions by clicking on the Generate Cooking Direction link
2. The system displays the Generate Cooking Direction page
3. The Web customer fills out the following required fields:
    - recipe label
    - recipe type
    - short description
    - recipe picture
    - Ingredients
    - Instructions
    - Preparation time
    - Cooking time
    - Portion
    - Calories
    - Price

*******Note: All fields above are required except recipe picture*******
The Web customer clicks on Generate Cooking Directions

#### 10.2.2 Edit data

11. The Web Customer elects to modify the information to be updated. The system displays the modifications as they occur.
12. The Web Customer clicks on the *Browse Recipes* link.
13. The Recipe site presents a search form with the following fields: recipe category, ingredients AND or OR Boolean operation on the provided ingredients to query for.
14. The Web Customer selects the search criteria using any of the fields the system provides.
15. The Web Customer elects to click on the “Search for recipe” button.
16. The Recipe site presents a list of recipes that have been previously created by a Web Customer.
17. The Web Customer clicks on the “Edit” icon on the recipe to edit the Recipe list.
18. The system displays the Edit form.
19. The Web Customer keys in the changes desired on any of the presented fields.
20. The Web Customer clicks on the <Submit Cooking Direction> button.

#### 10.2.3 Save data

The Web Customer requests to save the changes. The system validates that the values entered are appropriate to the type of data being edited. The system saves the information as pending changes upon successful validation. The system notifies the Web Customer of the successful save.

### 10.2.4 Use Case Ends

## 10.3 Alternate Flows

### 10.3.1 Handle Invalid data

This alternative flow begins in Save data step of basic flow when the system identifies that invalid data has been entered by the Web customer during the updating of recipe information.

1. Identify invalid data

The system encounters data inappropriate to the information being edited and prompts the Web customer to correct the entered data

2. Rejoin the Flow

The use case continues at the beginning of the Edit data step of basic flow.

### 10.3.2 Attach Recipe Picture

This alternative flow begins in Generate Cooking Direction step of the Basic Flow step when the web customer identifies the need to attach a photo during the Generate Cooking Direction Step.

1. Locate File

The Web customer elects to attach a file. The system prompts the Web customer to locate the file to be attached.

2. Attach File

The Web customer locates the file and elects to attach the located file. The system holds the file as an attachment to be added during the save.

3. Rejoin Flow

The system rejoins the Alternate Flow Generate Cooking Direction at step.

### 10.3.3 Handle Invalid File Format

This alternative flow begins in Generate Cooking Direction step of the Basic Flow step when the web customer identifies the need to attach a photo during the Generate Cooking Direction Step.

1. Identify Invalid file

The system encounters an incompatible file format during upload and prompts the Web customer to locate a different file.

2. Rejoin Flow

The system rejoins the alternate flow Generate Cooking Direction Page at Generate Cooking Direction step

### 10.3.4 Quit

This alternative flow begins at any point when the user identifies the need to exit the recipe Information.

1. Exit Recipe Information

The Web Customer elects to exit the Generate Cooking Direction Information. The system prompts to complete or cancel pending activities

2. Complete or Cancel Pending Activities

The Web Customer elects to either complete or cancel the pending activities. The system performs appropriate processing and notifies the Web Customer of success.

3. Use Case Ends

End of Use Case

## 10.4 Special Requirements

1. Editable recipe Information:

The system shall provide capability to edit the following:

- recipe label
- recipe type (e.g. lunch, breakfast)
- recipe description
- recipe ingredient
- recipe instruction
- preparation time
- cooking time

- portions
- calories
- price

### 10.5 Post Conditions

- Updated Recipe Information is in the system

### 10.6 Extension Points

N/A