Professor Razi Iqbal
Professor and Project Supervisor
Ontario Tech University

Dear Professor Iqbal,

The following report details the Software design and analysis of the “Clothify App” which is in development as an extension to the currently existing web application.

The purpose of this project is to provide another platform on which customers can shop at the “Clothify” brand, and which supports other technologies such as smartphones.

As recently joined System/Business Analysts and current Project Managers for this project, we have created a detailed report which provides detailed information in five phases of the software design for the app.

Each team member will also provide a detailed presentation to accompany each phase of the project.

We thank you for entrusting us with the responsibility of designing the “Clothify App” System and ensuring the company’s success. We look forward to further discussing the application.

Sincerely,

Mariyam Muhammad Alim;
Cristian Asprilla;
Sara Bhoira;
Chantel George;
Fareeha Malik and; Alagu Vallikkannan.

Project Managers
Clothify App

Software Analysis and Design for Clothify App

Prepared For

Professor Razi Iqbal

Prepared by

Mariyam Muhammad Alim

Cristian Asprilla

Sara Bhoira

Chantel George

Fareeha Malik

Alagu Vallikkannan

Business Analysts/Project Managers

Clothify App

April 4, 2023

Table of Contents

Phase 1:System Planning

Project Scope

The purpose of this project is to design and develop an online shopping experience where shopping is made convenient through an easy to use app. This mobile app available on iOS and Android allows customers to browse products and make purchases, track their existing purchases as well as add items to their favourites folder for later, and subscribe to the blog for exclusive opportunities and coupons.

Requirements

Functionality:

- Easy to use and strong user support
- Well-designed user interface for simplicity (sidebar with menu and search bar)
- Fast loading time (speed)
- Frequently updated (strong security)
- Excellent customer service (live chat)
- Outstanding image resolution

Usability:

- Allows customers shop with ease
- App provides effectiveness, efficiency, and satisfaction for customers
- Usability of the mobile app makes it simple for users to become used to the user interface
- Allows users to achieve their goal of purchasing item

Reliability:

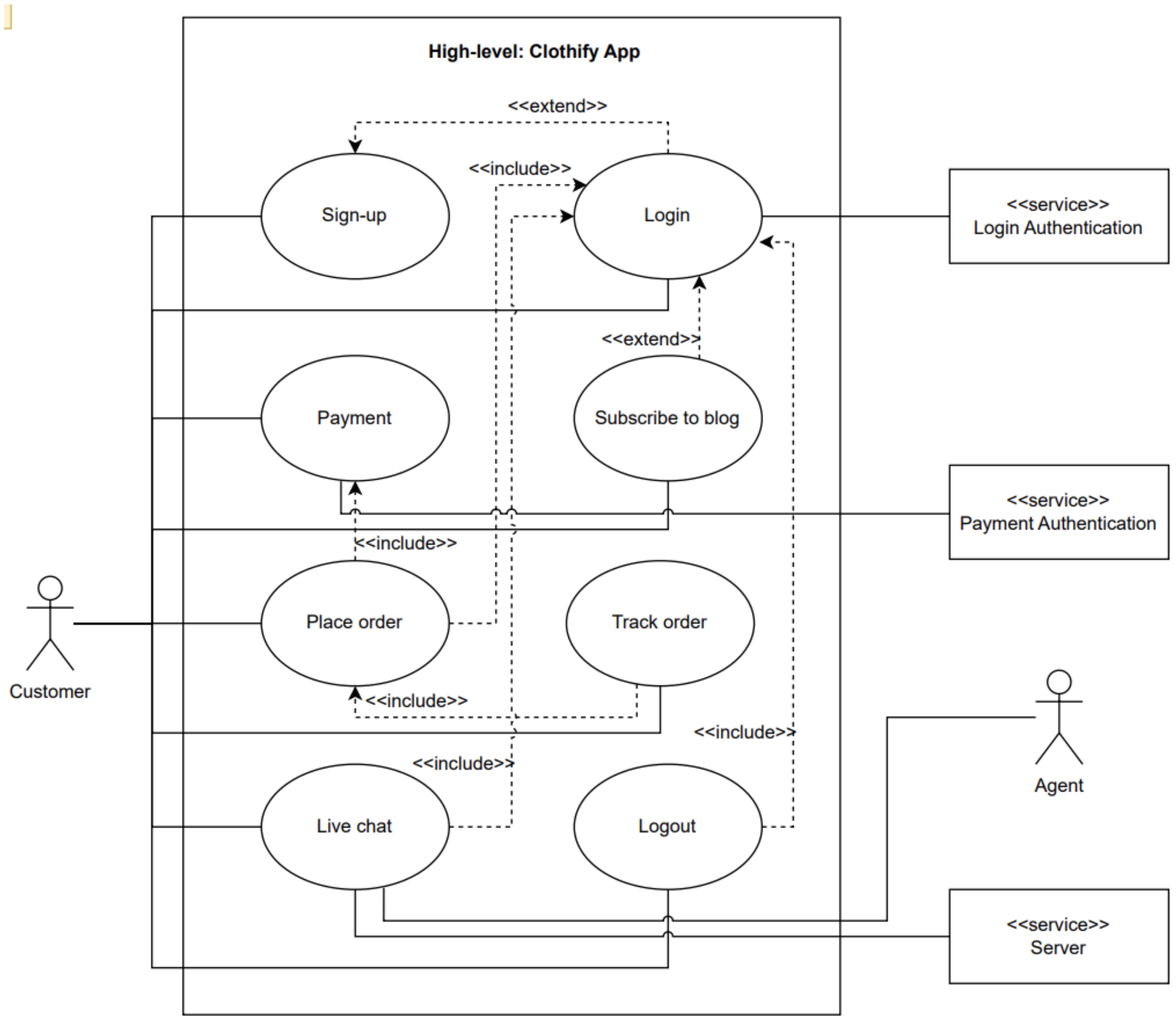
- Definitely meets the needs and expectations of the customers
- Frequently checking background system for any errors or scams
- Failure-free operation
- Provides high-speed performance
- Always ensuring the company is being productive by planning out resource usage

Supportability:

- Maintaining services such as the product catalog and live chat
- Always improving application for more efficient operations, and for high-level decisions
- App can support two languages, both English and French
- Maintaining security management and application evolution

Phase 2:Use Case Design

High-Level Diagram



The use case diagram above, is a high-level overview of Clothify app from the customers perspective. This diagram includes eight use cases that are, sign-up, login, place order, payment, track order, live chat, subscribe to blog, and logout. The actor in this diagram is the customer. The services for this diagram include Login Authentication, Payment Authentication, and Server. The following description will explain the use cases, and further explain the incorporation of the services. There is also a supporting actor included in this diagram which is the agent.

The sign-up use case is for the customer to create an account to enable access to the mobile "Clothify App" product page. Every time a customer opens the “Clothify App”, it will ask the user to login with their email and password. For that reason, the login use case is required for this diagram. There is an extend relationship from the login use case to the sign-up use case. This indicates that if it is a customer’s first time accessing the "Clothify App", creating an account is necessary before they may proceed to login. The Login use case has a login authentication service to identify that the correct credentials were entered by the user before proceeding to the "Clothify App"’s product page.

The Place order use case is for when the customer will browse through the app, select items, add items to their cart, and proceed to pay the amount for their order. The payment use case is required for the customer to purchase their items, by fulfilling a transaction payment. Here, the customer will also be able to select a type of shipping method for the order placed. A payment authentication service is connected to the payment use case to authorize if the transaction of the customer is valid.

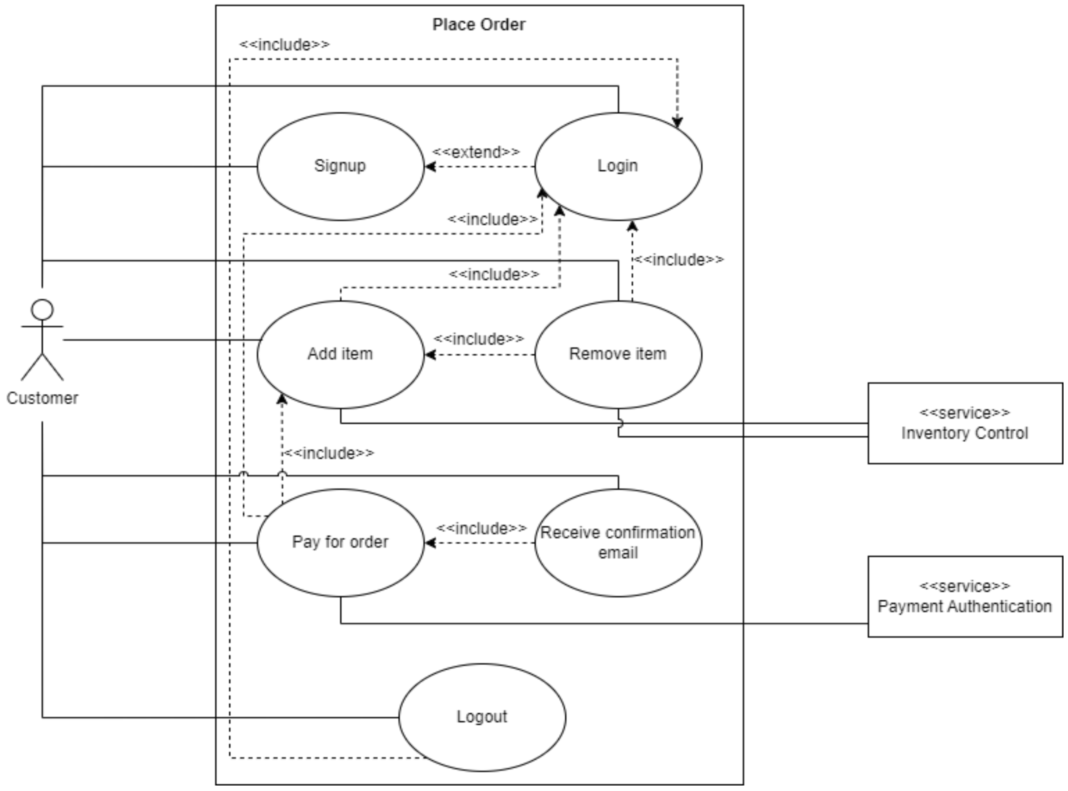
The live chat use case is a feature of the "Clothify App", where a customer can directly chat through online messages with an available agent. The customer can access this feature for any concerns or clarifications. For example, if their order is still not getting delivered when it is past the estimated date/time or an item is defective when delivered. For this reason, an agent (supporting actor) is connected to the Live chat use case. The live chat use case further has a server service for sending and receiving messages between the customer and agent.

The track order use case is necessary when the customer would like to see the status of their order. The status will provide the customer with information such as estimated delivery date and time. The subscribe to blog use case is needed for the customer to attain coupons or discount codes from the store. When the customer subscribes, they will receive a subscription.

From the diagram, there is an include arrow from the place order use case to the payment use case. This indicates that it is essential for a customer to place an order first before proceeding to the payment stage. There is also an include arrow from the track order use case to the place order use case. In order for a customer to track the status of their delivery, an order must be placed and paid-for.

An include arrow from the place order use case, the live chat use case, and from the logout use case are both directed towards the login use case as shown in the diagram. This indicates, in order for a customer to place an order, access the live chat, and logout, they are required to login with their credentials. Lastly, there is an extend arrow from the subscribe to blog use case to the login use case. This is to indicate that, in order for a customer to gain a subscription, they need to first provide their email and password. These are the same credentials used for logging in to the mobile "Clothify App".

Low-Level Diagram: Place Order

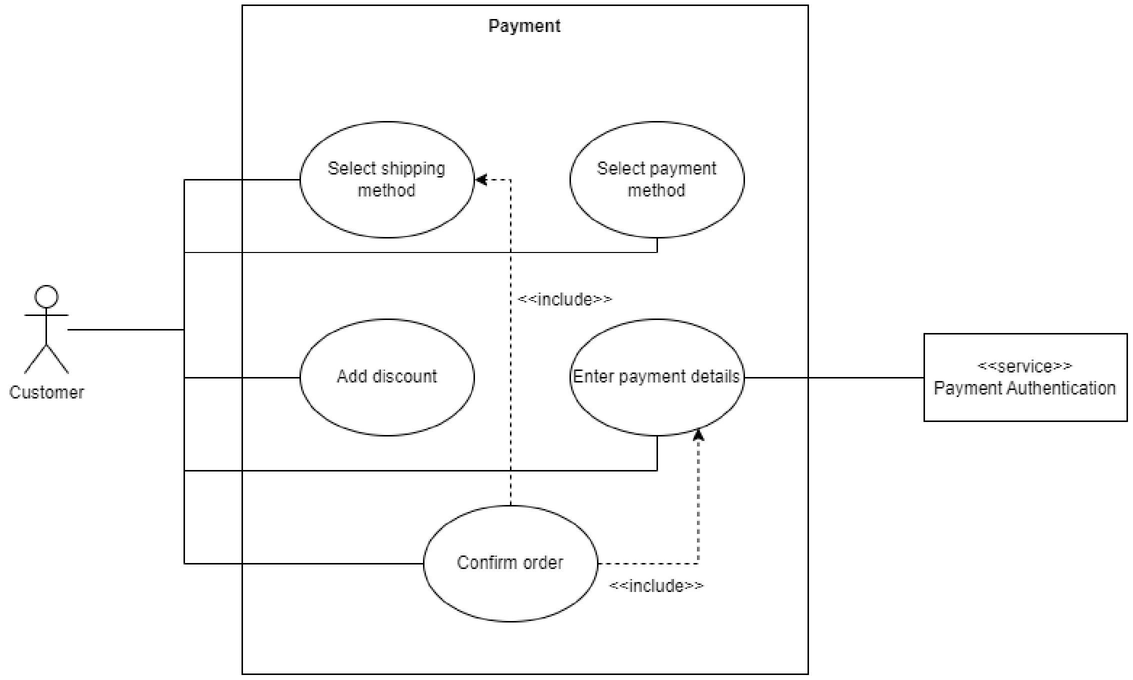


The diagram above shows the low-level use case for a customer placing an order through the app. There is one primary actor: the customer and two supporting services: inventory control and payment authentication.

The diagram shows all the use cases part of place order. In order to place an order, the customer needs to sign up. If the customer is already registered, they just need to Login which is why login extends sign up. Once they are logged in, they can add or remove items. Before the customer is able to add an item, the availability of that product is checked by the inventory control service. In order to remove an item, the user will have to have added an item. This is why the relationship between add and remove items is the “<<include>>” one.

The user can pay for an order only if they have logged in and added an item. The payment is verified by the payment authentication service. Once the payment is completed, the customer receives a confirmation email. Finally, the user can log out. Logout has to include login because the customer can not Logout without logging in.

Low-Level Diagram: Payment

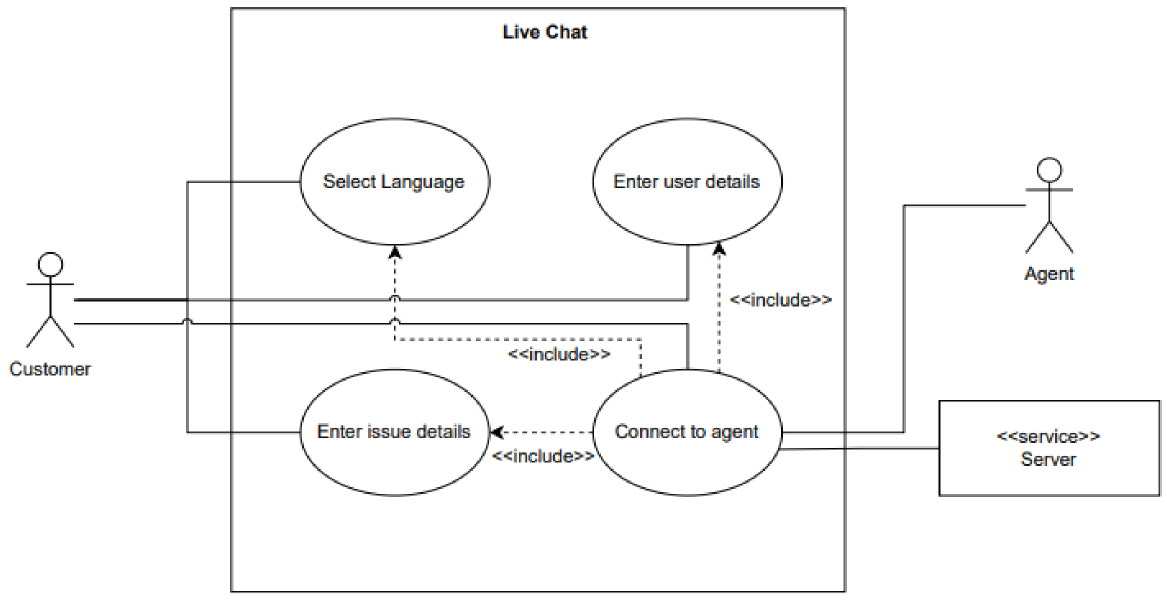


The diagram above shows the use case design diagram for the low-level use case “Payment”. In this diagram, there is one primary actor, “Customer”, one supporting actor, and a “Payment Authentication” service.

The diagram provides a visual understanding of how the customer can make a payment for their order. The customer can select their payment method. The customer can only confirm their order if they have entered their payment details and selected a shipping method, which is indicated by “<<include>>” relationships. Moreover, if the customer chooses, they may optionally add a discount to their order payment, as indicated by the “Add discount” use case.

A “Payment Authentication” service is also connected to the “Enter payment details use case” which acts as a third-party authentication service. Once the customer has entered their payment details and confirmed their order, then the third-party, such as a bank can authenticate the payment.

Low-Level Diagram: Live Chat



The diagram below shows the use case design diagram for the low-level use case “Live Chat”. In this diagram, there is one primary actor “Customer”, and two supporting actors one of which is the “Agent” and a “<<service>>” “Server”.

This diagram provides a visual understanding of how the customer can access the Live Chat functionality of the “Clothify App” in cases of issues. The customer must enter their issue details, and user details such as email, name and phone number and also select a language of correspondence before they can connect to an agent. Both the customer and agent are accessing the server which connects to the app.

Fully Dressed Use Cases

The diagrams below provide a detailed description of the app “Clothify App”, which acts as a separate interface to the current existing website on which customers can purchase items. One high-level diagram provides a textual description of the high-level design use case diagram and the remaining fully dressed use cases are textual descriptions for the three low-level use case diagrams chosen in the initial stages of the project.

High-Level Fully Dressed Use Case

The Fully Dressed Use Case represents the high-Level textual understanding of the high-Level Design Use Case diagram for the app. It should be noted that the app provides an easy-to-use User-Interface (UI) for customers. The customer can simply place the order and receive satisfactory, reliable service from the company. The only precondition for the customer to access the app is to have compatible technology to access the platform, such as a reliable internet connection or mobile data, a working and compatible device such as a mobile phone or computer and an email account to sign up and gain access to all app functionalities.

Name	Clothify App
Scope	Online application
Level	User Goal
Primary Actor	Customer
Stakeholders and Interests	● Customer: wants to place an order, requires it to be a reliable service, needs fast service ● Company: provide fast service to customer, wants to satisfy customer needs ● Agent: acts as a liaison between customer and company to solve customer issues
Preconditions	● Customer needs compatible technology to access app ● Email account
Success Guarantee	● Customer successfully places order ● Customer successfully receives order
Main Success Scenario	● Customer signs up or logs in to the app ● Customer adds items to cart ● Customer inputs payment details ● Customer confirms order
Extensions	● Customer inputs incorrect login information ● Customer denied access
Special Requirements	● Customer subscribes to blog ● Customer tracks order ● Customer causes live chat functionality

Frequency of Occurrence	Nearly continuous

Low-Level Fully Dressed Use Case: Place Order

Name	Place order
Scope	Online application
Level	User Goal
Primary Actor	Customer
Stakeholders and Interests	● Customer: wants to place order, requires it to be a reliable service, needs fast service with minimal effort ● Company: provide fast service to customer, wants to satisfy customer needs
Preconditions	Customer needs to sign up or login to the app
Success Guarantee	● Customer successfully places order ● Customer successfully receives order
Main Success Scenario	● Customer signs up or logs in to the app ● Customer adds items to cart ● Customer inputs payment details ● Customer confirms order
Extensions	● Customer inputs incorrect login information ● Customer denied access
Special Requirements	
Frequency of Occurrence	Nearly continuous

Low-Level Fully Dressed Use Case: Payment

Name	Payment
Scope	Online application
Level	User Goal
Primary Actor	Customer
Stakeholders and Interests	● Customer: wants to pay for order, needs fast service with minimal effort ● Company: store payment information ● Payment Authentication Service: verify payment information
Preconditions	Customer needs to enter valid payment details
Success Guarantee	● Customer successfully pays for order ● Payment authentication service successfully verifies payment
Main Success Scenario	● Customer enters shipping details (address) ● Customer selects shipping method ● Customer selects payment method ● Customer enters valid payment details ● Customer confirms order ● Payment authentication service verifies payment
Extensions	Electronic invoice sent to customer
Special Requirements	Customer can add discount code
Frequency of Occurrence	Nearly continuous

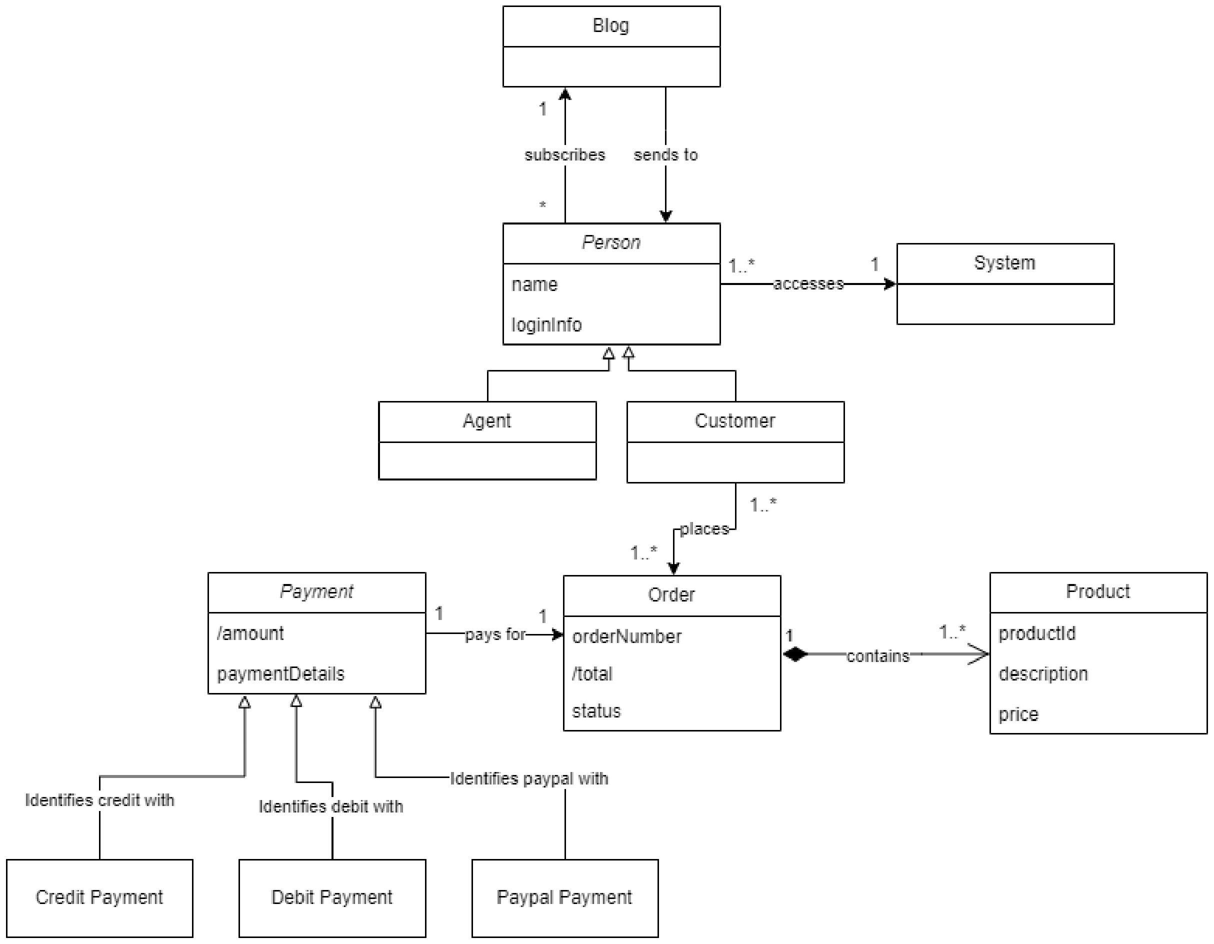
Low-Level Diagram Fully Dressed Use Case: Live Chat

Name	Live Chat
Scope	Online application
Level	User Goal
Primary Actor	Customer
Stakeholders and Interests	● Customer: wants to issue to be resolved with minimal effort ● Company: wants to resolve customer issues ● Agent: acts as intermediary between customer and company
Preconditions	Customer has an issue
Success Guarantee	● Customer issues are resolved ● Company gets good customer service feedback ● Agent successfully resolves customer issues
Main Success Scenario	● Customer enters issue description ● Customer submits request to chat with agent ● Customer is connected with agent ● Agent resolves customer issue
Extensions	Issue escalated to agent manager if unresolved
Special Requirements	Customer has service available in both English and French to resolve issue
Frequency of Occurrence	Nearly continuous

Phase 3: Domain Models

The domain model diagrams in this section build upon the high-level and low-level use cases identified in Phase 2. These diagrams will help highlight the main components of the "Clothify App" while also describing the relationship between them. They will serve as a foundation for all the other phases during the design of the application.

High-Level Diagram



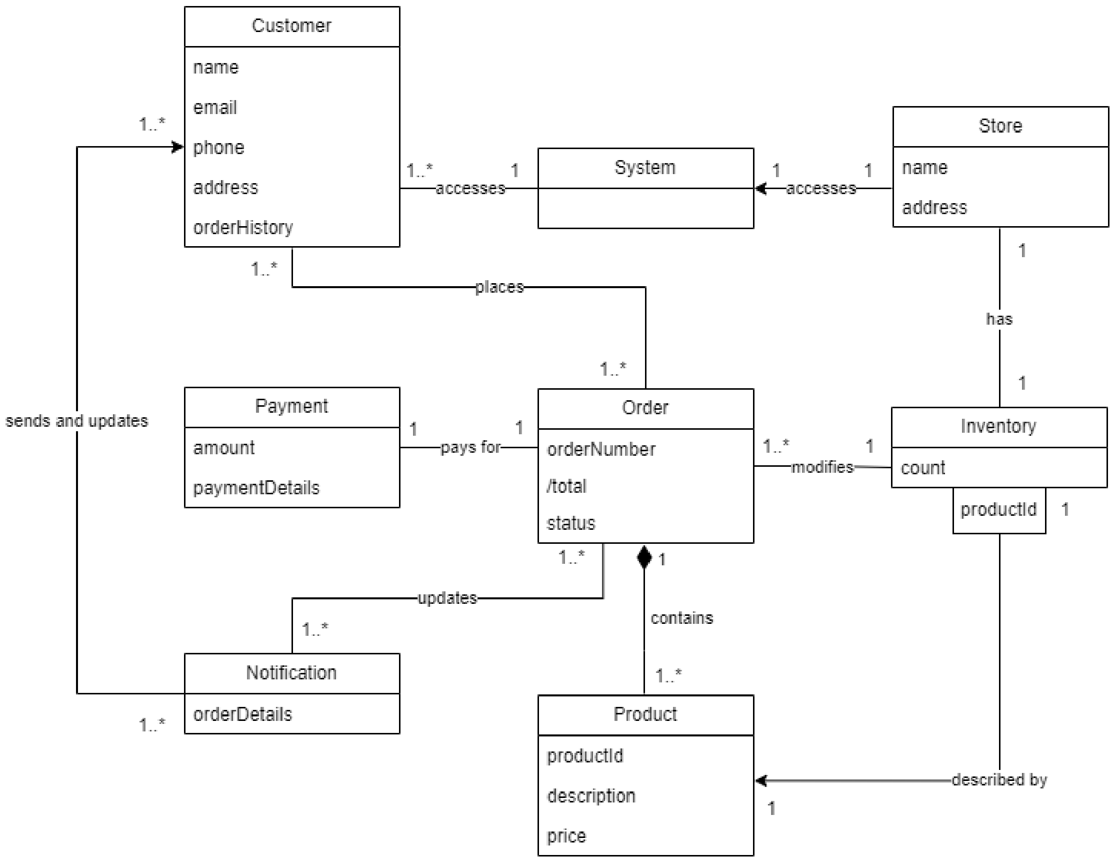
The following high-level domain model diagram provides a general overview of the classes required to fulfill the functionalities of the application identified in phase 2.

First, the Person class, is one of the most important entities, since it represents the different types of users that interact with the system. In this case, there are two subclasses derived from the class, Customer and Agent. Another important class is the System, since it allows communication between the different parts of the application (system) by allowing users to access it.

On the other hand, the Order class represents the orders that a customer has, and the Product class represents the products that are being ordered (an order can have multiple products). There is a Payment class that helps pay for an order. This class contains three subclasses that represent the different types of payment methods that customers can use to pay for the order. These are Credit Card, Debit or Paypal.

Finally, there is a Blog class. Customers can subscribe to the blog and when they do, they are eligible to receive discount codes.

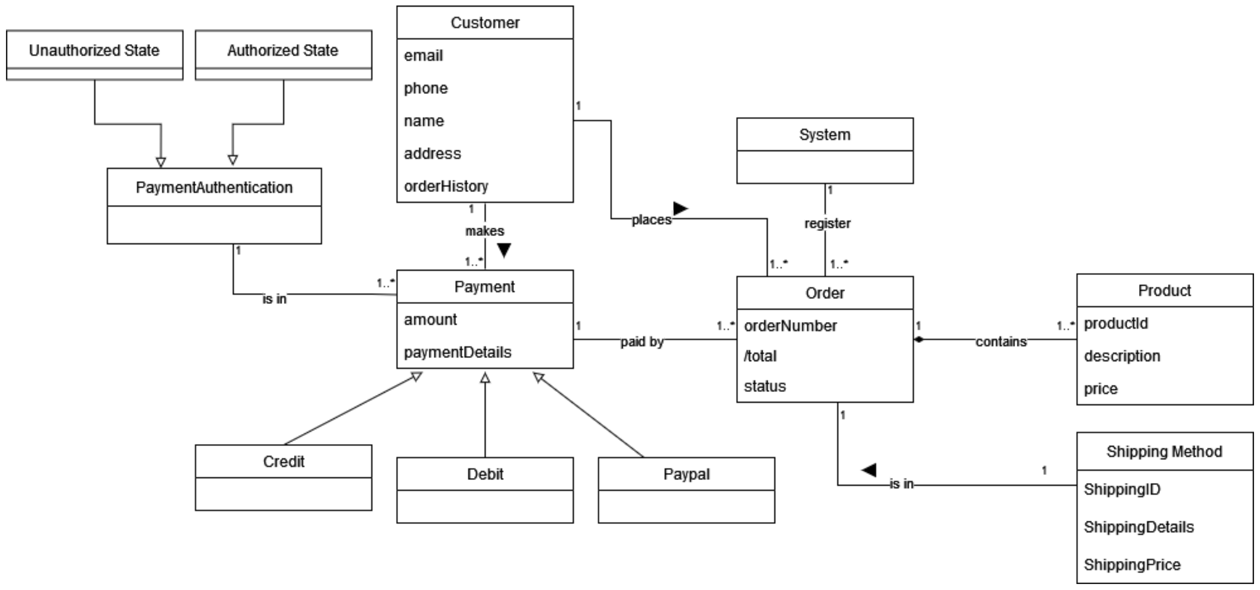
Low-Level Diagram: Place Order



The low-level generalized domain model diagram for place order below consists of several key components. There is one system that can be accessed by both the store and its customers. The Customer class holds information about customers, including their order history. The store has an inventory of products which can be described by the Product class. The Inventory class keeps track of product availability. Customers can place orders through the app, which is represented by the Order class. An order may contain multiple products, and it can be paid for with one payment. Once an order is placed, the status is changed to "Order confirmed," and the Notification class is updated with the order details. The Notification class then sends a confirmation email to the customer. The Order class would also modify the counts of products in the Inventory class once the order is confirmed.

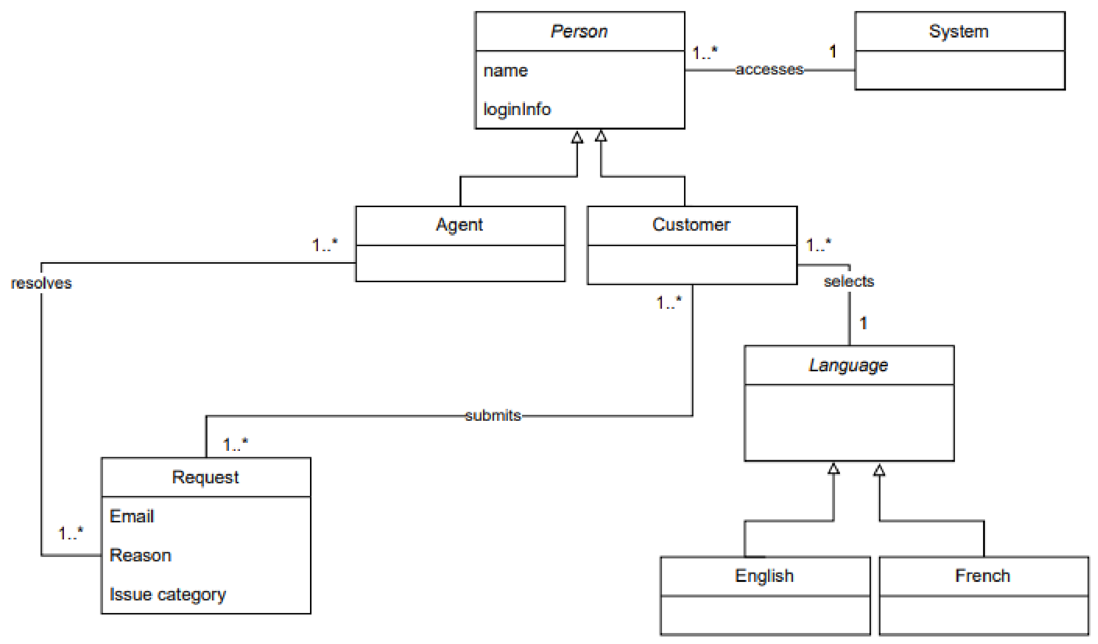
These design decisions were made to ensure the system is accessible to both the store and customers and to provide an efficient and effective way to place orders and track their status. By incorporating these components and their relationships, the domain model can serve as a useful guide for the development of the system.

Low-Level Diagram: Payment



This diagram represents the main components of the payment section and the interaction between the different classes. The customer class contains the attributes necessary for the process of making payment for an order. These are email address, phone number name address and order history. The Order class contains the order number, the total to pay for that order and the status of that order. The product class contains a product id that allows the identification of each product, a description for this product and the price of that product the payment class contains the amount of the payment and other payment details, the payment can be made by credit card, debit card or PayPal. There is a payment authentication class which makes the validation of the payment that the customer makes and in turn reflects whether the payment is authorized or unauthorized and the system class is the main entity that allows registering the orders in it and everything that carries the details of those orders. On the other hand, the relationships of this diagram are that a customer can make one or many orders, in turn, an order or many orders are paid by a payment method, a customer can make one or many payments, in addition as mentioned above all orders are recorded in the system, and the system can record one or many orders, an order contains one or many products, in turn, the shipping method is part of an order, and an order can have only one shipping method this shipping method contains a shipping identifier that allows tracking of this shipment and a shipping price that is added to the order total.

Low-Level Diagram: Live Chat



The diagram below represents the low-level Domain Model diagram for the Live Chat use case. This Domain Model Diagram was created through the help of the Phase 2 Use Case Diagram and Fully Dressed Use Cases for Live Chat.

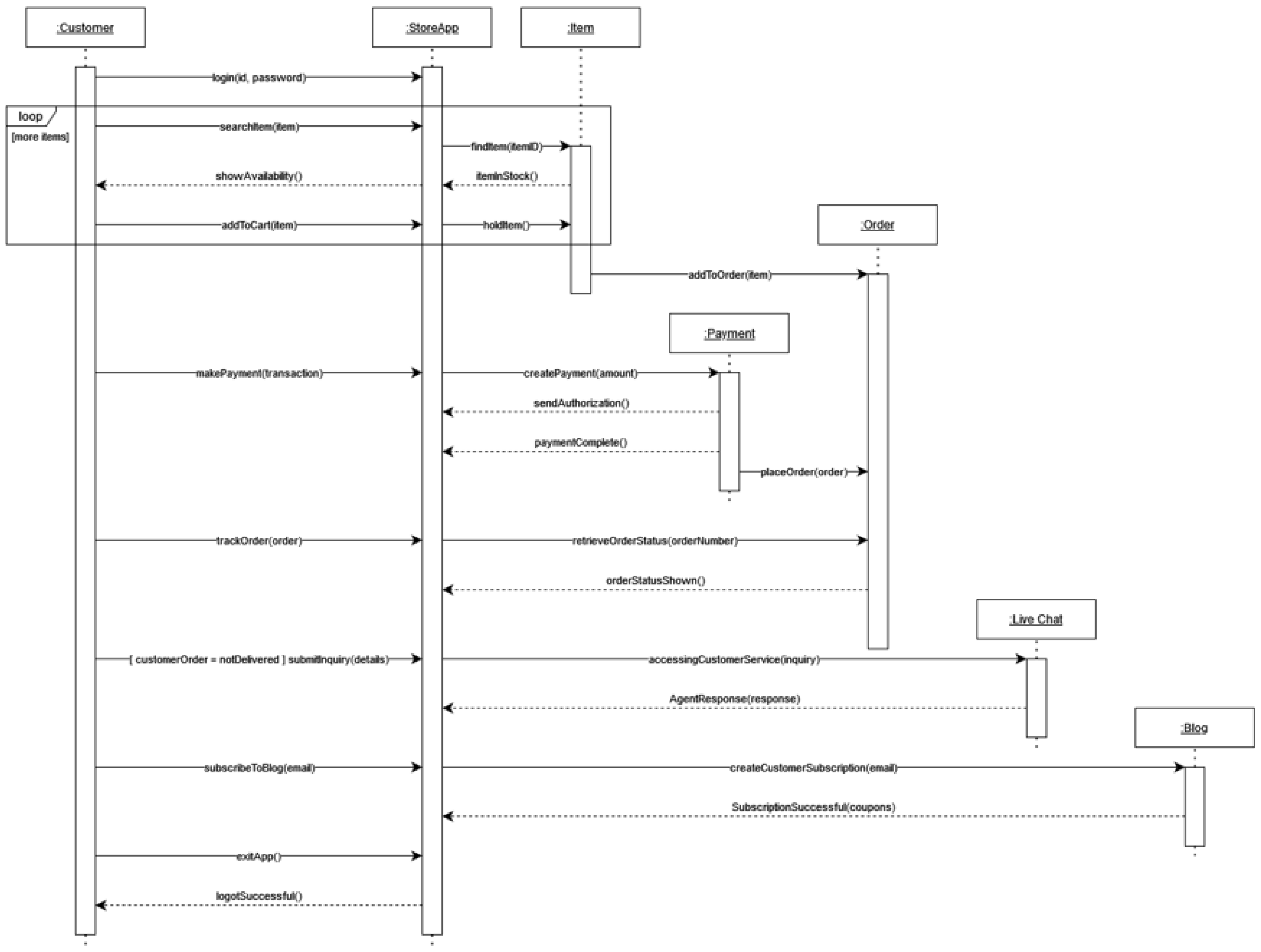
It should be noted that A “Person” class exists with “Agent” and “Customer” sub-classes. The “Person” class accesses the system, on which the “Clothify App” and other functionalities such as Live Chat would be stored.

Similarly, a “Language” class exists. From this class, there are subclasses of “English” and “French”. The purpose of this class is to show the relationship between the “Customer” and the “Language” classes that allows the customer an option to choose between the two languages for correspondence while having their issue resolved by the agent.

Furthermore, the “Request” class includes Customer information that would be essential to an issue they file for the Agent to resolve. This would include “Email”, “Reason” and “Issue Category”.

Phase 4: Interaction Diagrams

High-Level Diagram



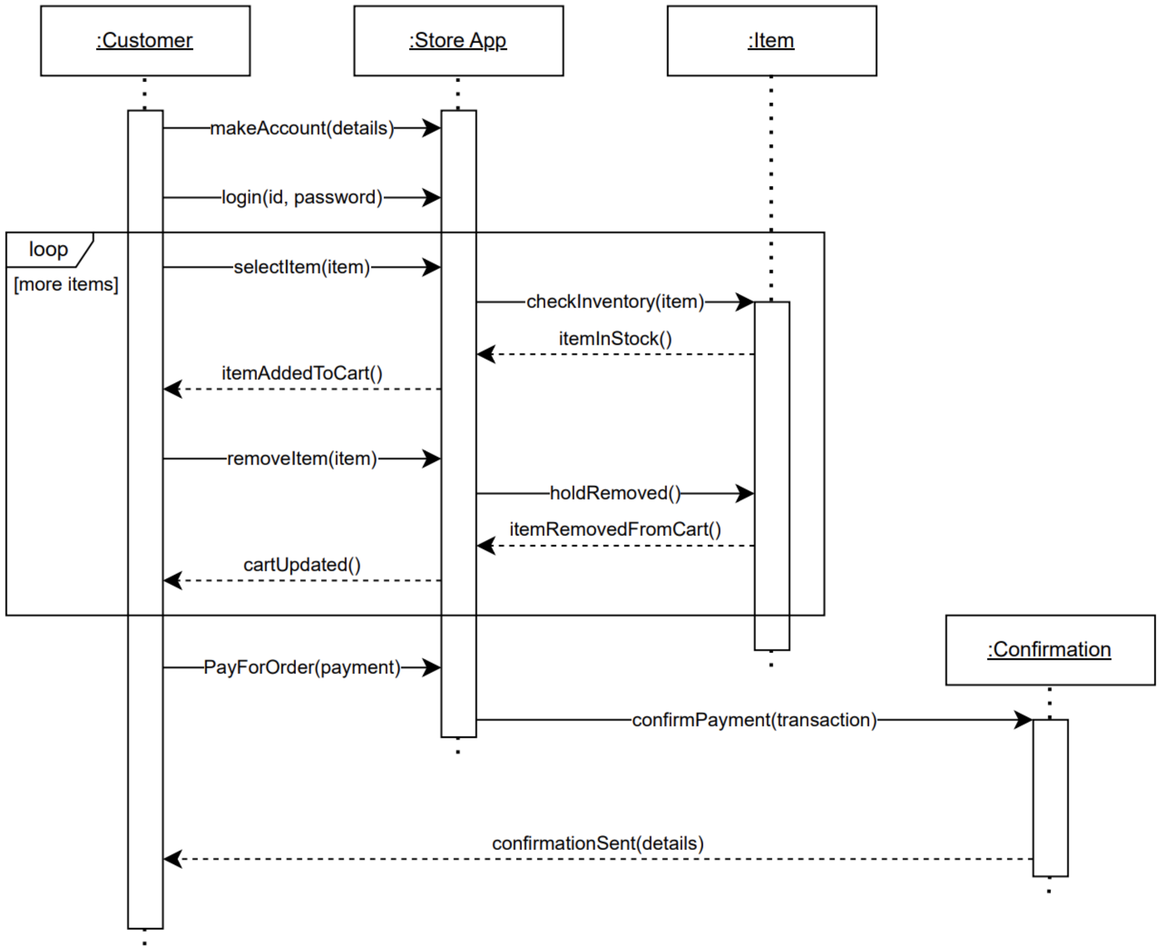
The interaction diagram above, is a high-level overview of the “Clothigy App” from the customer's perspective. This diagram includes 7 classes that are, customer, "Clothify App", item, payment, order, live chat, and blog. The following description will guide the customer through the flow of the diagram, starting from the top. In order for the customer to browse through a variety of items and make a purchase on the mobile app, the customer has to first log in with the user details. After entering the id(username) and password, the customer will be redirected to the “Clothify App” product catalogue.

The customer may now begin to scroll through the app for items that fit the customer's needs and preferences. The customer will search for an item, by browsing through the listed options, and then view a particular item by selecting it for a more detailed description of the item and its availability (number of items currently in stock). In the background, the "Clothify App" will find the item (through the ID), checking if the requested item is in stock. The "Clothify App" will then display the results for the customer. These results will include the number of items that are available in the inventory. If the customer is satisfied with the item and it is currently in stock, the customer can add this item to the online shopping cart. The "Clothify App" will then place a hold on this item. The customer can continue to search for items to the customer's preference, and add more items to the cart. This is conveyed from the loop drawn on the diagram for more items.

Once the customer is satisfied with the cart, the customer can proceed to make the payment transaction. The "Clothify App" should then receive authorization for the payment to be completed. After the payment is completed, the order will be placed for the customer. The customer can then track the order through the "Clothify App" by providing the order number. The "Clothify App" will then be able to retrieve the customer's order status and display the results.

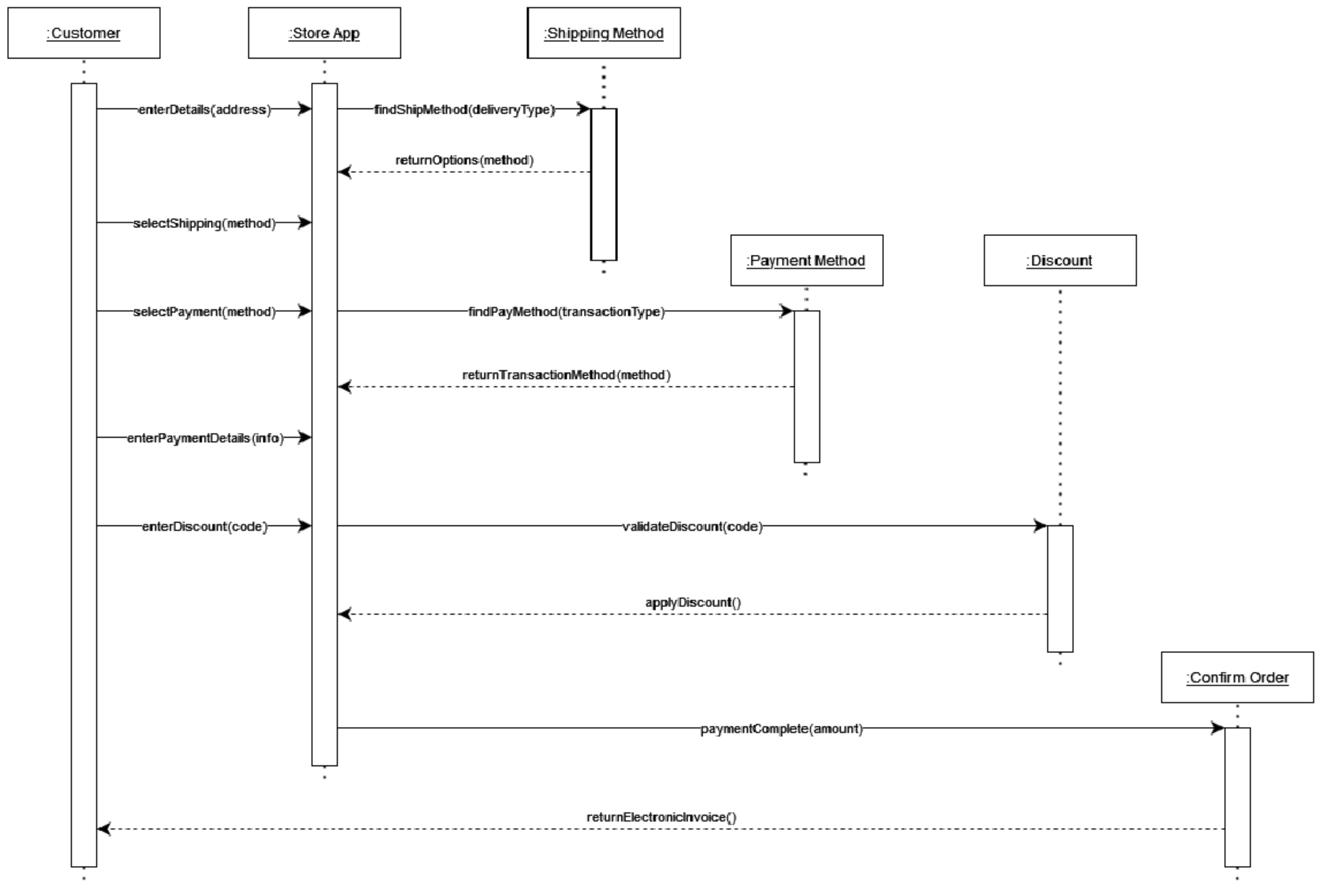
If the customer has any concerns, such as the order not being delivered within the designated time, the customer may use the live chat feature on the "Clothify App". This feature enables a customer to chat with an agent through online messages directly. Once the customer has submitted an inquiry, with details of the customer's concern, an agent will respond with a solution. The customer also has the option to subscribe to the "Clothify App" blog. This is a subscription the customer would receive that will contain coupons or discount codes. The customer may exit the app by choice.

Low-Level Diagram: Place Order



The diagram above shows a low level interaction diagram for customers placing an order through the mobile app. As shown in the diagram, a Customer and StoreApp instance exists, where most of the associations occur. For example, the StoreApp would be able to access the details the customer enters in the app like the customer's personal information, specifically the name, phone number and email address. The loop around the arrow selectItem(item) to the cartUpdated() is a significant feature of the interaction diagram because the customer can purchase many items in one order rather than having multiple orders. Once the items are selected, the customer would pay for the order by entering the payment credentials, which would then be accessed by the StoreApp. The confirmation instance would confirm the payment transaction received by the StoreApp and would send the confirmation details to the customer.

Low-Level Diagram: Payment

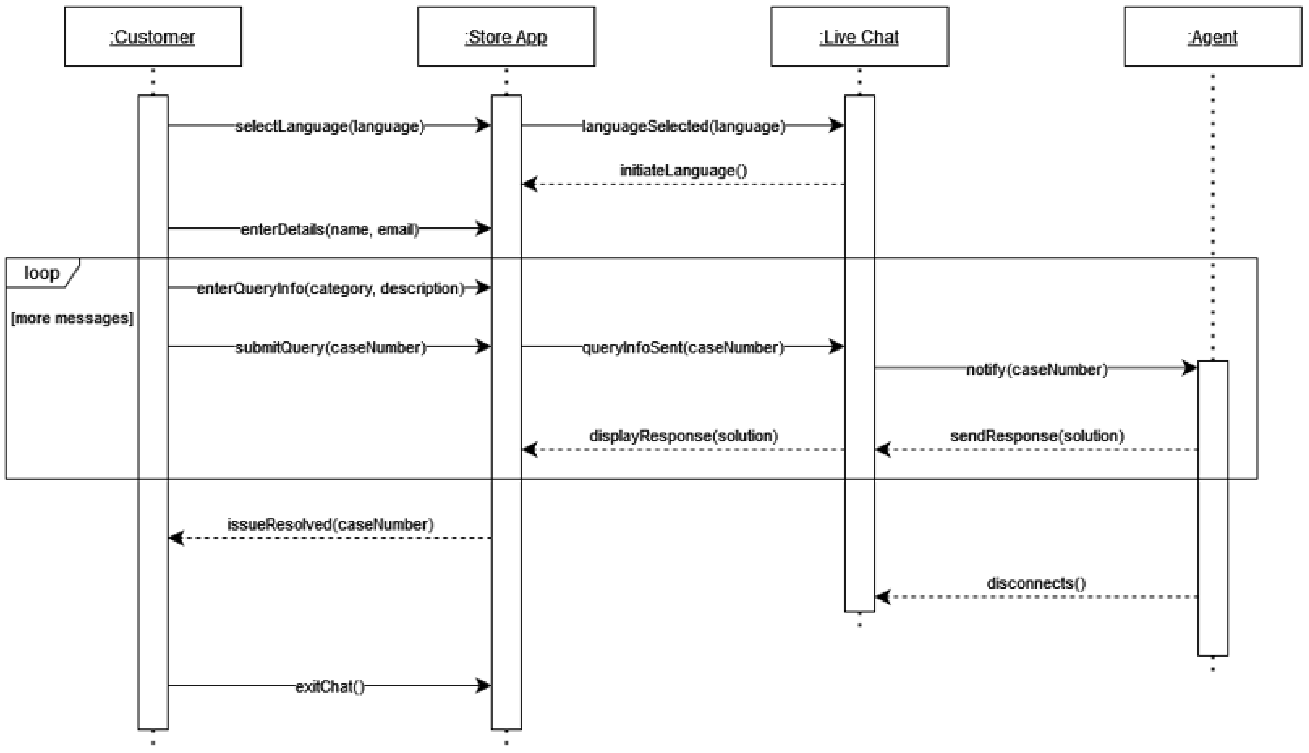


This diagram displays a low level interaction diagram to make a payment for the order in the customer's existing cart. This diagram as shown contains 6 classes, Customer, "Clothify App", Shipping Method, Payment Method, Discount and Confirm Order. To begin the payment process, it is required for the customer to have an updated shopping cart and to have gone through the place order process as a previous step.

The diagram clearly indicates how every Customer interaction goes through the "Clothify App" instance to fulfill every request. The payment process begins with the customer entering the shipping details to the app, which then requests the Shipping method class to find available shipping options for the specific customer depending on the address. Once the preferred method of shipping is chosen, the customer gets an option to select the preferred method of payment. Once chosen, the Payment Method instance returns the transaction method with a secure payment form containing the required details for this method of payment, which the customer enters into the app. If the customer has a discount coupon, an option to to enter the code is available, which is then validated by the Discount class and applied once the discount is considered valid.

With the modified total after the discount, the payment is completed through the Payment Method class and sent to the Confirm Order class where the Electronic Invoice is created containing the amount and other payment information.

Low-Level Diagram: Live Chat



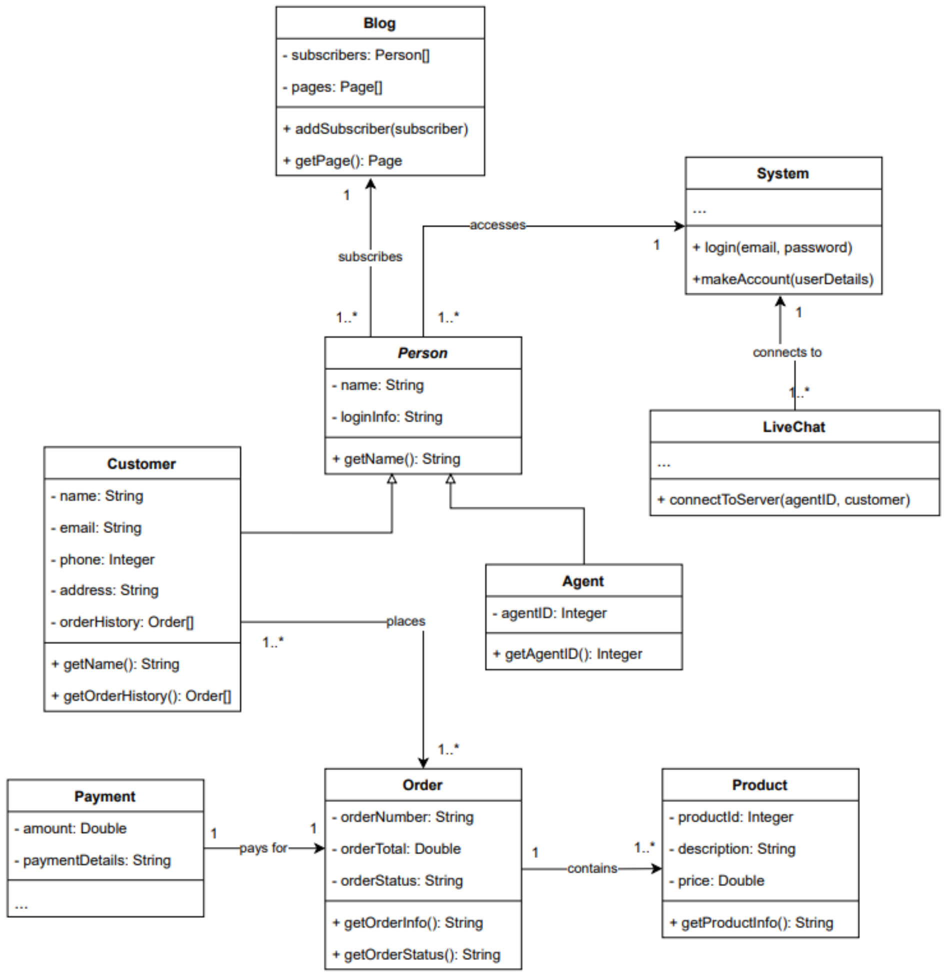
The interaction diagram above, is a low-level overview of the live chat feature in the mobile "Clothify App". This diagram is shown from the customer's point of view. The diagram includes 4 classes that are, customer, "Clothify App", live chat, and agent. The following description will guide the customer through the flow of the diagram, starting from the top. When the customer selects the live chat feature on the app, they will be prompted to select a language, that is either English or French. Once the language is selected, the "Clothify App" will redirect the customer to the live chat.

The customer will then enter the user details, such as name and email. They will then enter the query info, that will include the category of the customer’s concern along with a description, and then submit. The customer’s message will then be sent, notifying an available agent. The agent will send a response back to the customer with a solution that will be displayed for the customer on the live chat. The customer and agent may continue to send and receive messages on the live chat until the issue is resolved. This is shown from the loop drawn on the diagram for more messages. When the customer’s concern is resolved, the agent will disconnect, being available to assist the next customer. The customer may then exit the app.

Phase 5: Design Class Diagrams

The Design Class diagrams in Phase 5 follow the structure and format of the Phase 3 Use Case Design diagrams. They provide a more detailed understanding of how the “Clothify App” software would look like.

High-Level Diagram



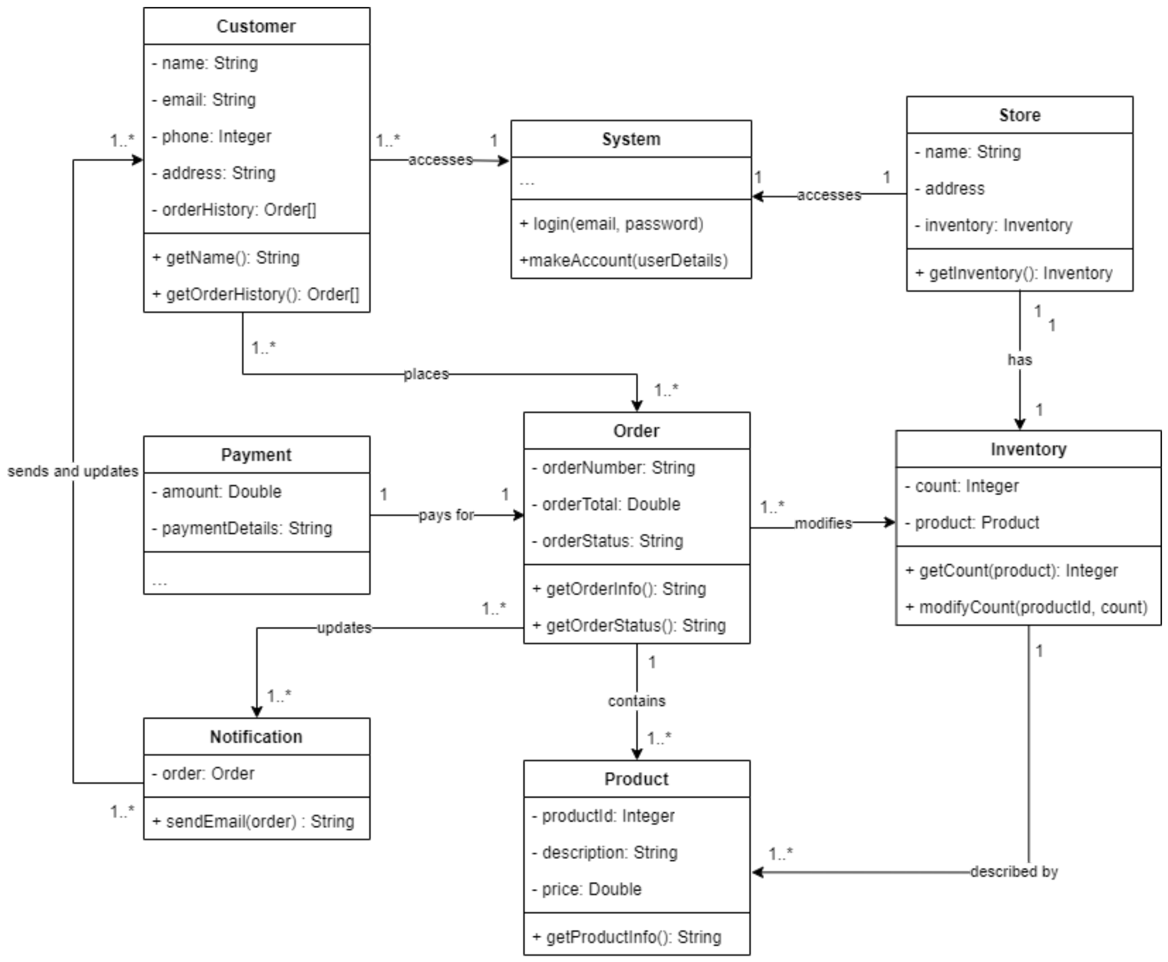
The high-level design class diagram below focuses on the classes that relate to the functionalities of the app. The Person class includes private name and loginInfo String variables which can only be accessed by the specific Person Class. A getter method is also present for the system to “get” the customer name information.

It should be noted that in the Blog class, there is a Person array which will store an array of subscribers to the Blog system. This is a private variable so only the Blog class can access this information. Similarly, a public “pages” variable exists which is stored within the Page array, to store an array of all the pages a customer can browse on the blog. Essentially it acts as an individual database and system.

Furthermore, the “System” class acts as a connection between the “Person” class which includes “Customer” and “Agent”. Due to the “System” class existing, there is a “LiveChat” class which is a functionality within the app for “Customer” and “Agent” to communicate through.

All of the classes in the diagram below have meaningful, self-explanatory names, with the relevant methods, variables and getters. However, not every class includes the getters in the methods, as it is understood that they would be included in the software development process.

Low-Level Diagram: Place Order



The class diagram below provides a low-level design of the classes that were identified for place order in Phase 3 of the project.

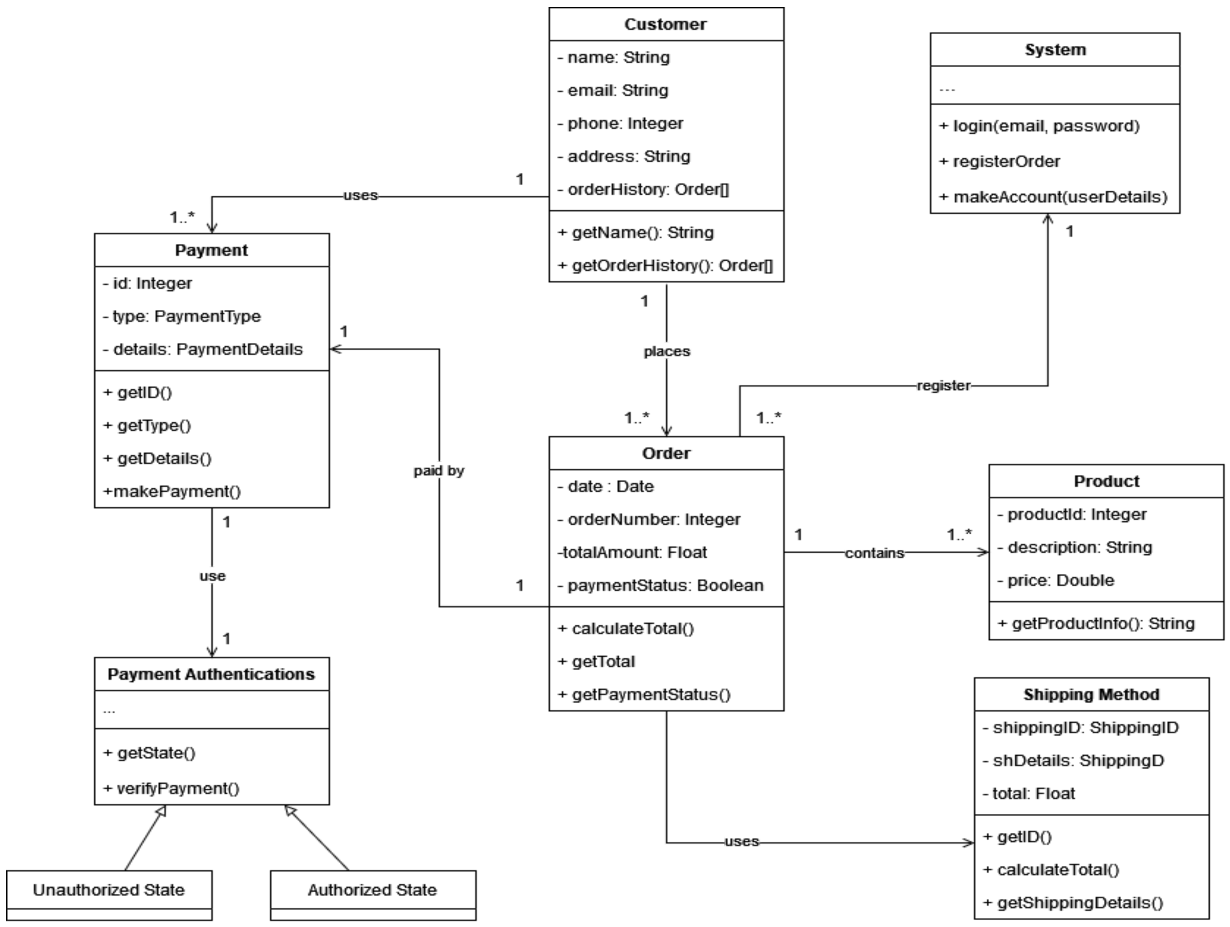
The Customer class, along with all other classes, has getters and setters for all its data attributes. Only a few of them are shown in the diagram. The System class has a login method that allows customers to Login to the app. It also has a makeAccount method for new users.

Additionally, the class stores user details, store records, and other private information, which have not been shown in the diagram as the method of storage is yet to be decided. The methods for the Payment class have not been included either (see low-level diagram for payment).

The Inventory class has a modify count method which allows other classes such as the Order class to modify the count of specific products when an order is placed.

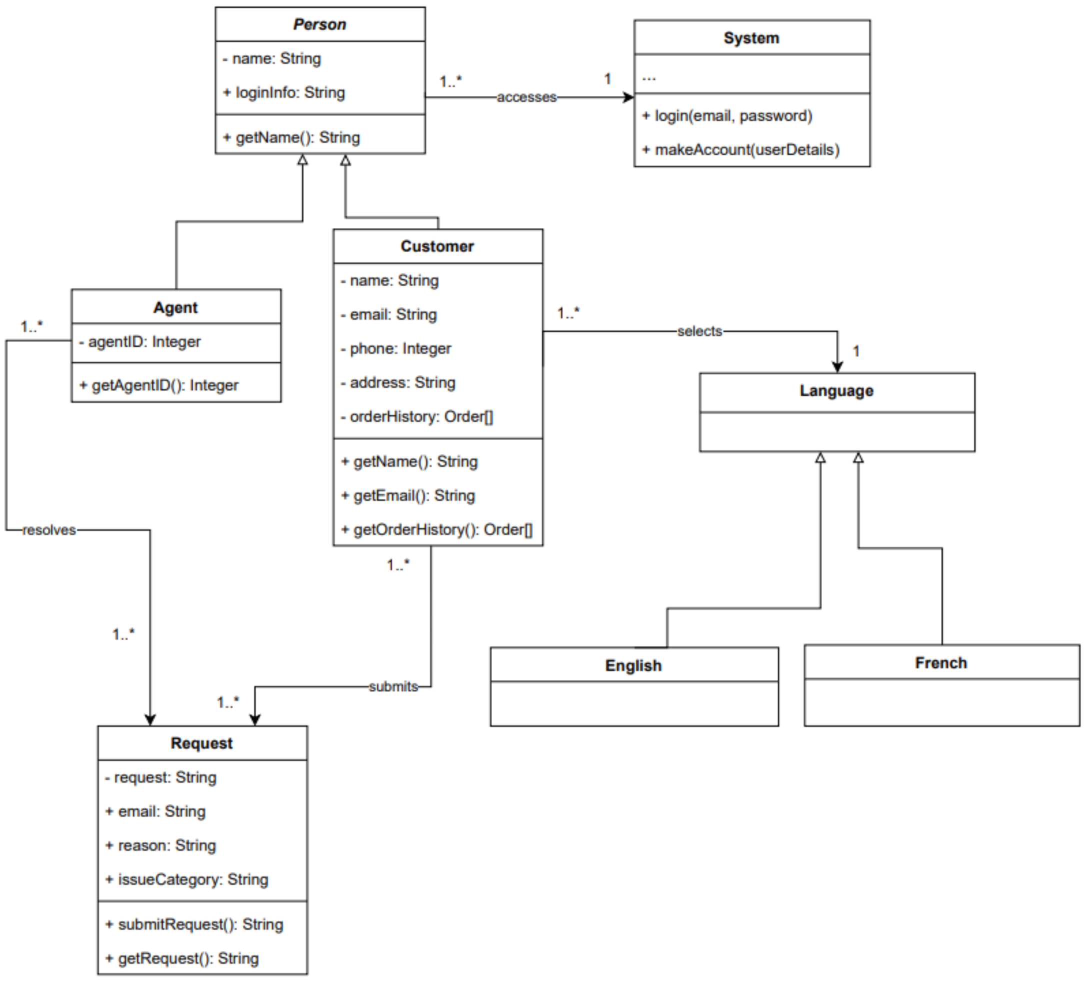
Finally, when an order is confirmed, the Notification class is updated and sends a notification to the customer while also updating their order history, as described in Phase 3.

Low-Level Diagram: Payment



This class diagram is for the payment process, it is similar to the domain model only that in this case will be defined the main attributes and methods of each class; for example the Customer class has the private attributes name, email, address, and orderHistory as well as their respective methods to get and set these values, the Order class contains the attributes date that indicates the date on which the order was placed ,the orderNumber, totalAmount and the paymentStatus of this order, a method to calculate the total for this order. The Product class contains a productId for each product, the product description and the price, the Shipping class contains the attributes shippingID, shippingStatus and total. On the other hand, PaymentAuthorization class contains the attributes ID the type of payment, and the payment details, a function called to makePayment() that allows the customer to use this to make the payment, and finally the System class have functions login() that allows the customer to log in to the application, registerOrders() that allows the registration of each order that occurs within the system or application and makeAccount() that allows the creation of accounts to the new user are stored. It is necessary to highlight that all the classes that contain private attributes have the respective set and get methods to obtain access to this data from other parts of the system.

Low-Level Diagram: Live Chat



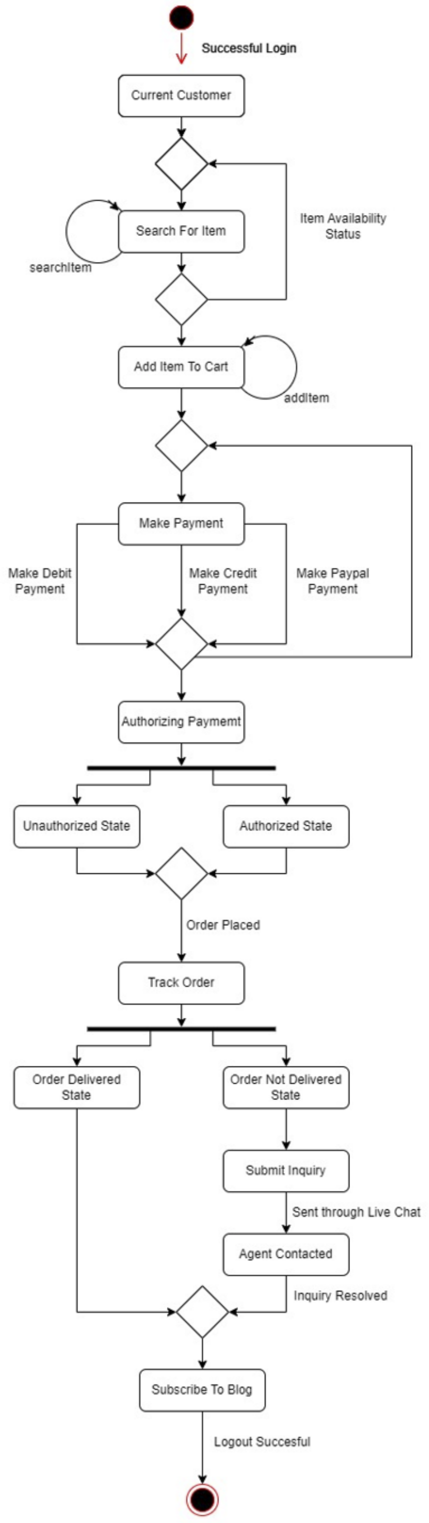
The diagram below represents the low-level Design Class diagram for the Live Chat use case. This diagram acts as a detailed extension to the Phase 3 Low-Level Live Chat diagram.

It should be noted that not every class includes the relevant getter methods related to their private variables, as it is understood that they would be included in the software development process.

One notable class and its variables include is the “Request” class which is more detailed in this phase. It now contains private and public variables and the necessary methods to allow the Customer to submit the request “submitRequest()” and for the Agent to be able to access the request through the “getRequest()” method.

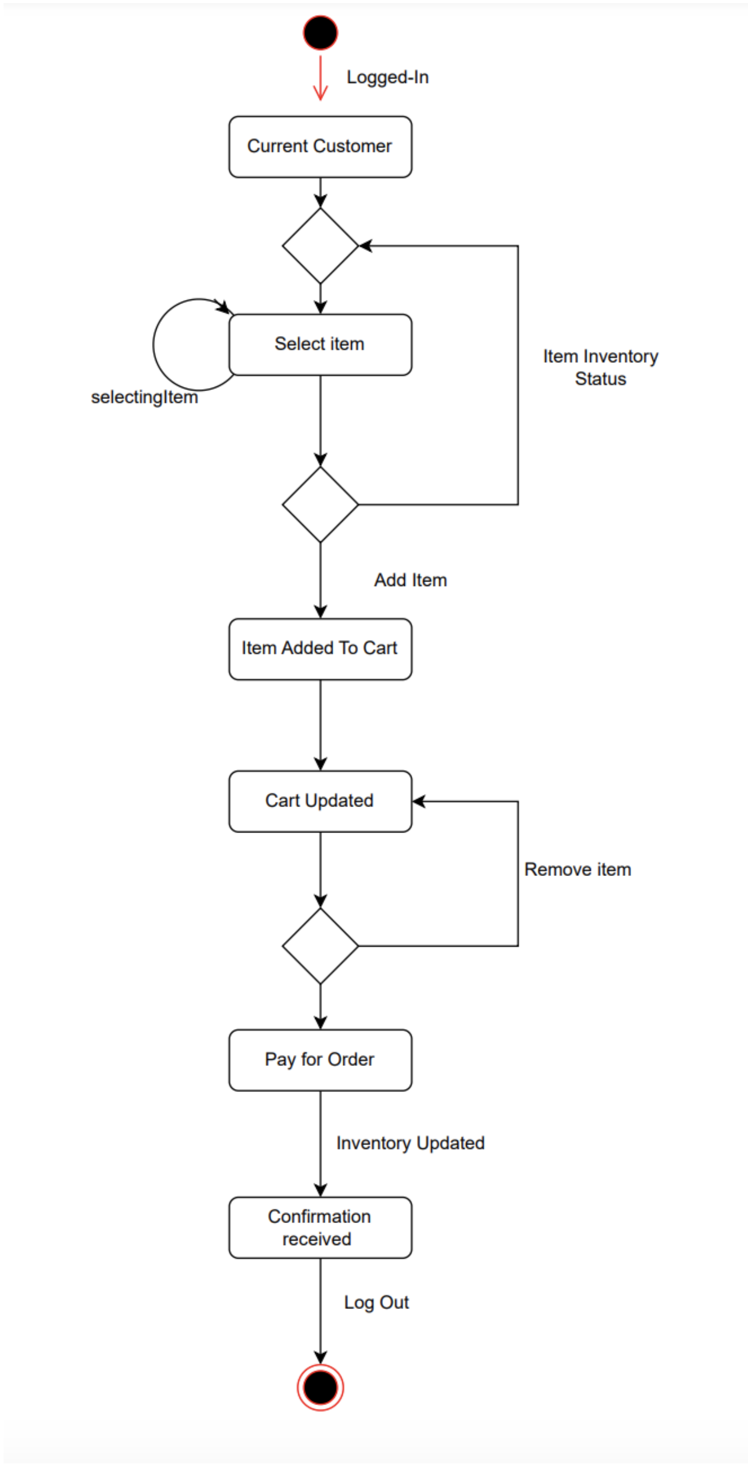
Phase 6: State and Package Diagrams

High-Level Diagram



The diagram above displays a high-level overview of the state diagram that describes how a customer can place and track an order. The system has many exciting features to help customers learn about the procedure and how the app works. When the app is downloaded, the customer is required to login because if the login is unsuccessful, the customer wouldn't be allowed to choose the items. Once the customer is logged in, the customer will be able to search for the favourite items and add them to the cart. In the Search for Item and Add Item To Cart states, there’s a loop to show that the steps can occur multiple times throughout the process of placing the order. After all the items are added to the cart, the customer will be moved to the payment state, where the system will be asked to enter the payment details. The customer can select one of the three payment options shown in the diagram, which are debit, credit, and PayPal methods. If the customer successfully completes the payment process, the order will be placed and will move to the Track Order state, but if the payment is unsuccessful, it would go back to the payment state, since the order has not been processed yet. Once the payment state has been completed successfully, the customer will be able to track the order which is one of the exciting attributes the app has. In this state, the customer would be able to see where the order is at the moment. The other amazing feature the app has is that the customer can subscribe to the blog and have a conversation with an agent to resolve any inquiries the customer may have. Once the order has been delivered to the client, the client can subscribe to the blog to receive coupons and any offers the app has. If the order hasn’t been delivered to the customer, the customer can send an inquiry through the live chat about the issue the customer is facing. When the issue has been resolved, the customer can move to the Subscribe to Blog state and log out the app once everything is done.

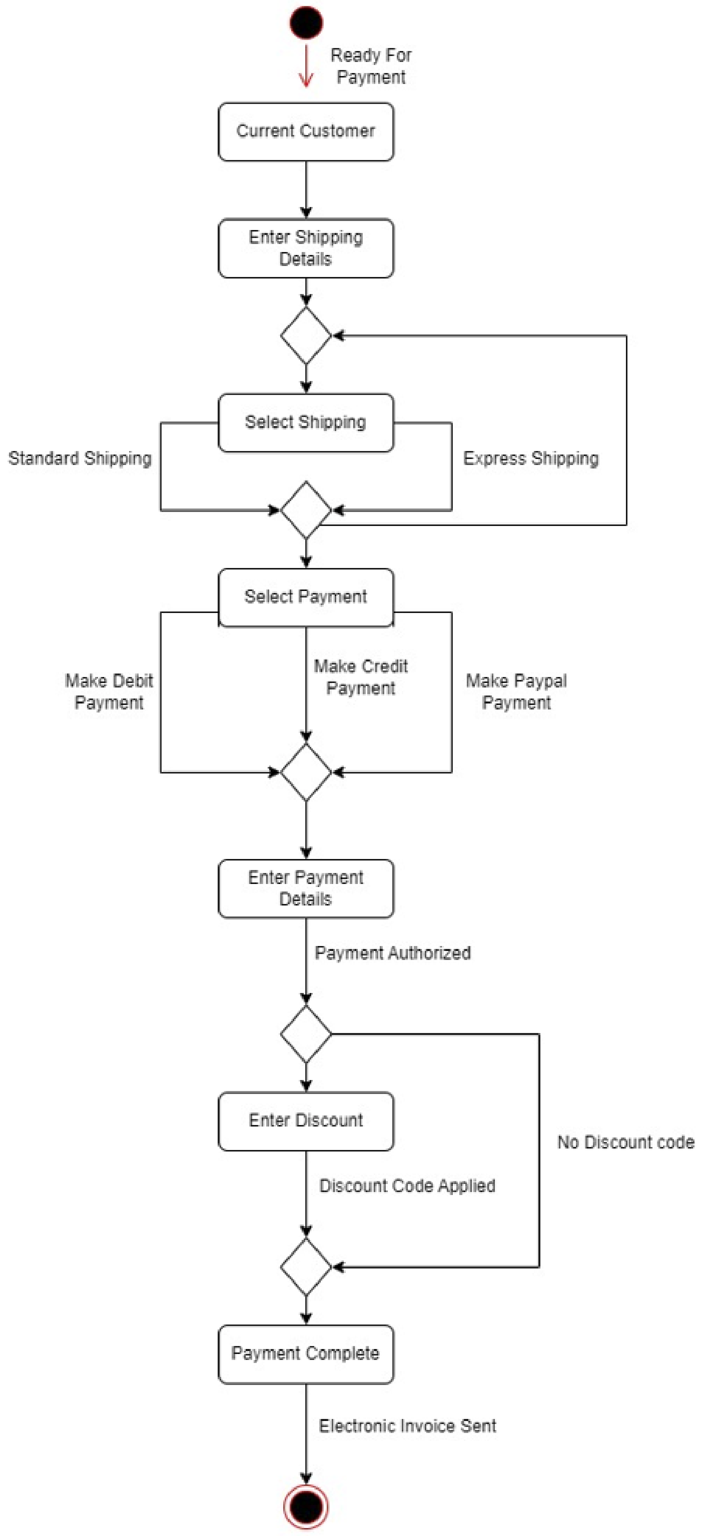
Low-Level Diagram: Place Order



This diagram displays a low-level state diagram of a customer's process of placing an order. As indicated in the diagram, the requirement to begin this process is to ensure that the customer is already logged into the app, prior to reaching the select item state.

The Select Item state is shown in the diagram as a loop since the customer has the choice of picking more than one item to place an order. Once the item has been selected, there is a decision node to see whether the item is in stock or not. If the item is in stock, the customer is directed to add the item to the cart. In the case of the item not being in stock, the customer is redirected to pick a different item. Once the item has been added to the cart, the cart is updated with the new number of items, as well as the new estimated total for the order so far. Following the cart updated state, there is a decision node which represents that the customer also gets the option to remove an item once they have at least one item. If an item is being removed from the cart, the process is back in the Cart Updated state, and the customer gets to either keep removing more items or continue by paying for the order. Once the process has proceeded from the Pay for Order state, the store inventory is updated. While in the Confirmation received state, the customer receives the Order Confirmation containing information about the items the order contains as well as the information from shipping to payment and personal details. Once this confirmation has been received, the customer either manually logs out, or exits the app which gives the same outcome to end the place order process.

Low-Level Diagram: Payment

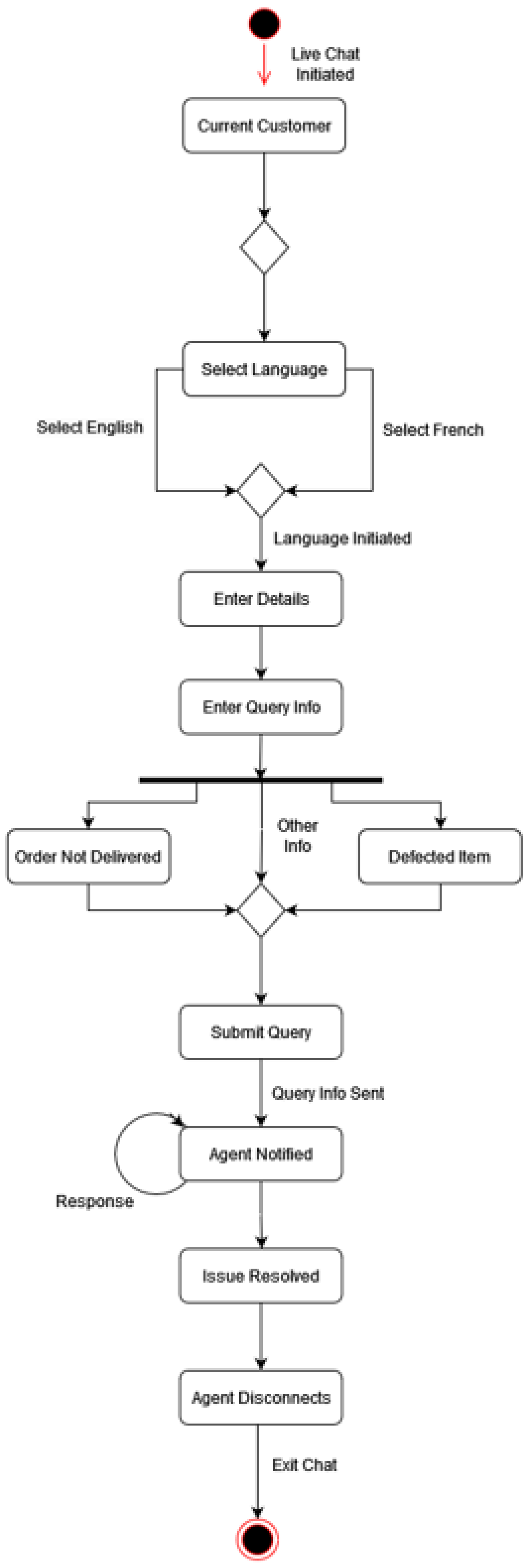


The state diagram above, is a low-level overview of the payment process in the mobile "Clothify App". This diagram is shown from the customer's point of view. The diagram includes 6 states that will be thoroughly explained. The following description will guide the customer through the flow of the diagram, starting from the top. When the customer is ready for payment, the first state will be to enter shipping details, such as name and address. Then the customer will decide which type of shipping method to select for the order placed. The customer will be presented with two options to select from, either standard shipping or express shipping.

As one can view in the diagram above, there is a decision node before and after the select shipping state. The arrow from the second decision node to the first decision node indicates, the customer can re-select the shipping method of choice. Once the customer completes selecting the choice of shipping method, the customer will proceed to the select payment state. Here the customer is presented with 3 different transaction types that include, Debit payment, Credit payment, or PayPal payment.

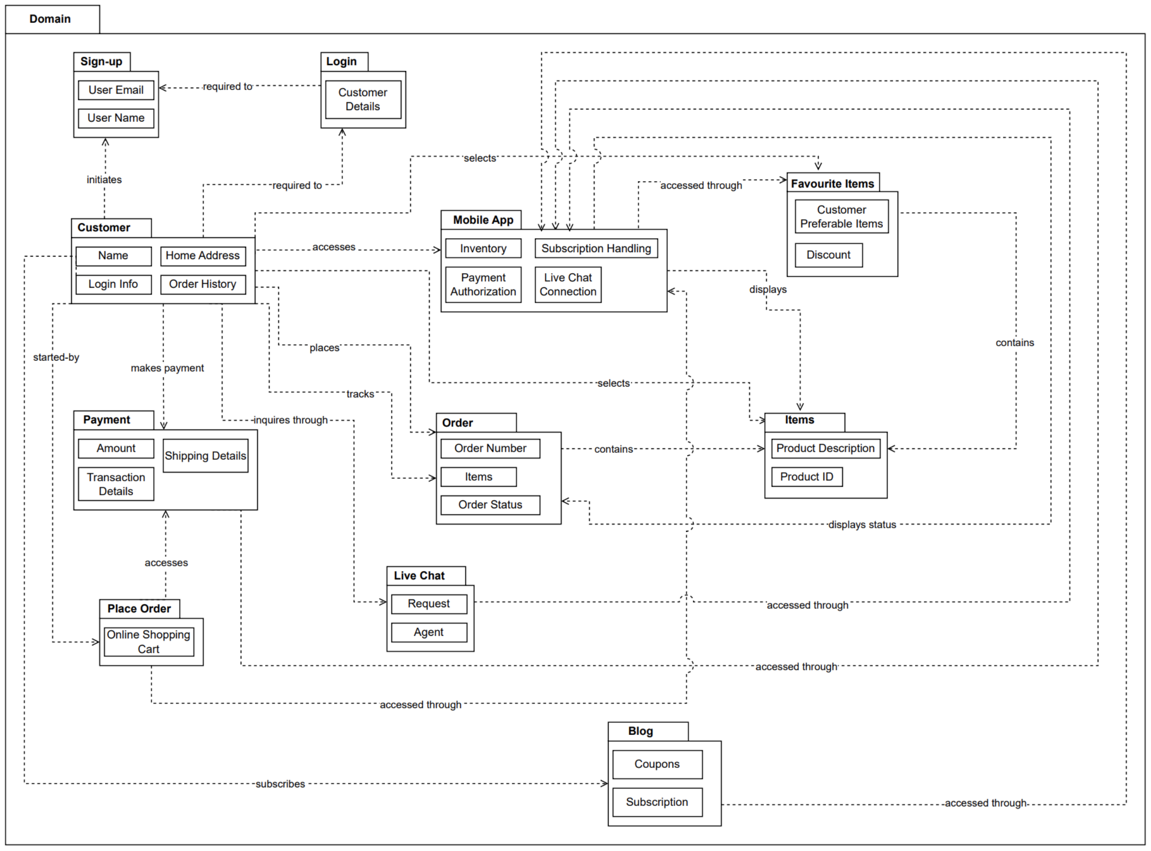
After the payment method is selected, the customer will continue to the payment details state, where the customer will enter information such as the card number. The payment details should get authorized. Following that, the customer will be directed to enter the discount state, where the customer can enter a code for a reduction in the total amount to be paid for the order. If the customer has no discount code currently available, the customer may move onto the next state. This is indicated from the arrow starting at the fourth decision node, and directing towards the fifth decision node. Finally, the customer will proceed to the payment complete state where the customer will receive an electronic invoice after completing the transaction for the order.

Low-Level Diagram: Live Chat



The diagram above represents a low-level state diagram for customers who want to chat with a live agent about their inquiries. The system would ask the customer what language they prefer to be communicated in, French or English, as both are common languages in Canada. Once the customer selects a language, the system asks the customer to enter the details and queries in the chosen language. As displayed in the diagram, the possible reasons the customer would like to talk to the agent can be because of a defective item, items not being delivered, or some other reasons such as items not fulfilling the required needs and any discounts available in the app. The agent would receive the query submitted by the customer and have a conversation with them to see how the customer can solve the issue. Once the agent solves the customer's inquiry, the agent gets disconnected from the server, and the customer exits from the chat.

Package Diagram



The diagram above displays the high-level overview of a package diagram for this app. This diagram contains 11 packages, all of them connecting to or from the Customer package. The Customer package, containing Name, Home address, Login info, and Order History, is what accesses the Mobile app, initiates sign up and requires to be logged in, selects items or favourites, makes payment, tracks and places orders, subscribes to Blog, and inquires through the live chat. The “Clothify” Mobile App, containing Inventory, Subscription Handling, Payment Authorization and Live Chat Connection, is the main source of communication between the Customer and the rest of the package. Packages such as the blog, payment, live chat, and the orders are accessed through and items are displayed on the Mobile App. Sign Up Package includes the User Email and Password, which is initiated by the Customer and required to access the Login class that contains the Customer Details. The Items package contains Product Description and Product ID, displayed through the Mobile App, can be detected in the Order and Favourite Items. The Favourite Items package contains the Customer’s preferable items as well as Discounts, and can be accessed through the Mobile App. The Payment Package contains the amount, shipping details, and transaction details, and is accessed by the order through the Mobile App. The Order Package contains the items, is tracked and placed by the Customer, and accessed through the Mobile App. Place Order Package results from the order package and contains the Customer shopping cart, ready to be placed on order. The Blog Package contains Coupons and provides a subscription for the customer, and accessed through the App. Finally, the Live Chat Package contains the customers Request and the Agent, is accessed through the Mobile App, and inquired by the Customer.