Iteration 1: Write Up

Team Members: Hayley Rymer and Teme Woldesenbet

Use Case Domain Diagram

An actor must be linked to a use case in any use case diagram. With this particular diagram, the actor is linked, but several use cases are not linked to any actor. This allows for more complexity in the code. The argument can be made that once a trip has been created, several different routes can be taken to reach the final use case or the Thank You Note indicating that the trip was not only taken but was successful. This particular diagram shows the simplicity of the software, which is ideal for any user experience. Their process should be streamed online inefficiently, just like this diagram. The actor, Premium clients are connected to Premium, and the use case Thank You Note displays a circular notation of cause and effect. The diagram provides valuable insights into the Premium e-commerce platform. It enables the development team to better understand user requirements, design the system's object-oriented structure, and visualize the system's behavior over time.

State Machine Diagram

Because a state machine diagram portrays the software’s behavior; this diagram shows intricacy despite the minimalism. There will be multiple objects created for this program. For example, Once a reservation has been made, the state changes to "Payment Pending" until the customer's payment has been received and accepted. If the payment is accepted, the state changes to "Paid/Active" and the itinerary is generated and sent to the customer. If the customer cancels the reservation, the state changes to "Cancelled". If the trip is completed as scheduled, the state changes to "Completed". If the customer is refunded, the state changes to "Refunded".

The diagram only shows essential objects that are implemented to make clarification these are dynamic options and will produce the correct user interface to move forward in the program.

Class Diagram

The class diagram for the program shows the risks of the software. Users are associated with a need to keep interfaces simple. The diagram shows the one-way street to plan a trip. There need to be multiple avenues to allow users to move in any direction throughout the software. Users may only want to complete a step to move to a new step or page. For example, some users want to check out payment options before booking a trip. The software must allow access to all screens and not restrict any flows. Users who get frustrated and are not provided immediate feedback are less likely to continue using the software and recommend it to others. However, this diagram has potential risks that should be addressed to ensure a comprehensive understanding of the system.