# Cross-Carrier Flight Reservation System

By Beth Richardson and Gabe Eapen

## Contents

# 1. The Domain Problem and Stakeholders

## 1.1 Domain Description

The goal of this project is to create a Web-based travel agency that provides airline ticket price and schedule information across multiple carriers. Travelers can view and filter comparisons between providers based on price, length of travel, and schedule. Travelers can then make reservations directly through the system and then review and modify those reservations that have been made through the system. Additionally, travelers expect to be able to share those reservations with others traveling with them. In some circumstances, such as when a travel administrator or coordinator is making reservations, travel plans may also be made by a planner, acting on behalf of the actual traveler.

Travelers increasingly want to plan, execute, and review travel reservations in one system, which is available from all of their devices. They also expect to be able to manage all aspects of that trip within a user interface with an intuitive format and streamlined processes. The vision for this system is geared toward end users with little to no formal travel booking experience. Because the application gains revenue through successful travel reservations, the reservation process should be a natural and fluid process, with as little chance for frustration as possible.

## 1.2 Envisioned System

Our vision for our system has three top priorities: availability, reliability, and usability. Because our system is for a travel system, it must always be available to travelers, regardless of where they are and what time they need access to their travel plans. Travelers also rely on this system to ensure their travel plans are not inadvertently cancelled, made incorrectly, or exposed to third parties. Finally, because the system will be accessed by all different types of users, it must be easy and intuitive to use so that our site is used over other travel providers or directly through the carriers.

Our application must be available on as many devices and platforms as possible and without server downtime, so that travelers can make, review, and update their travel reservations at any time while on vacation as well as at home and in the office. This requirement leads us to a web-service based architecture that utilizes existing APIs for flight information and reservations and exposes its own API for accessing reservation accounts and reservations.

Although we need a universally available interface, the infrastructure must also maintain secure and reliable access to data. Servers should ensure that reservation and client data is never lost or fail to book reservations as expected. Additionally, client account security must be maintained to protect sensitive payment, travel, and identity information. Encryption and token-based authentication will be a part of the system to ensure that identity is maintained, while data is not able to be accessed beyond the verified user, regardless of which web service is being accessed or which part of the site the user is accessing.

## 1.3 Stakeholders for the Architecture

### 1.3.1 Consumers

1. Travelers
    a. **Expectations**: The application should be highly available to provide immediate access when planning, booking, and managing reservations. It should provide one location to compare across many flight options and always have reliable data and secure transactions. Account information should be private and only shared with those with whom the user directly shares.
    b. **Role with respect to development project**: End user
    c. **Role with respect to domain**: Travel consumer
    d. **Part of organization they work**: Any part
    e. **Impact**: The project will fail if the travelers' expectations are not met. All revenue relies on travelers being satisfied with their use of the application, recommending the application, and continuing to use the application.
2. Traveler Planners
    a. **Expectations**: Planners have the same expectations as Travelers, but also have additional expectations related to managing other travelers' reservation information. Planners expect to be able to always access shared traveler data in an intuitive way. Planners expect to not accidentally book travel for their own account when booking for other users. Planners expect to only be shared appropriate data based on correct sharing privileges.
    b. **Role with respect to development project**: End user
    c. **Role with respect to domain**: Travel coordinator
    d. **Part of organization they work**: Most likely Executive Assistant or family member to a Traveler
    e. **Impact**: The project is prone to customer support complaints and potential lost revenue if Planners expectations are not met. Overtime, this will lead to project failure as Planners who are unhappy with the application will not return as Travelers in the future.
3. Airline Carriers
    a. **Expectations**: Should ensure that reservations are made correctly. Should ensure that fees are collected correctly and transmitted. Should ensure that all data presented to and collected from travelers matches exactly with what is transmitted from and to the carrier systems.
    b. **Role with respect to development project**: Carrier experts, partners
    c. **Role with respect to domain**: Manage airline expectations

    d. **Part of organization they work**: Airline carrier customer agents, carrier finance managers
    e. **Impact**: If carriers have issues with our system, they may request that we do not make reservations on their systems in the future or may recommend to Travelers not to use our system. It is critical that we provide a seamless interface for the Airline carriers so that the Traveler's expectations are also met.
4. Travelers' Friends
    a. **Expectations**: Can access and view shared trip information. They should only be able to see the trip details that are directly shared with them. They should see accurate data for those trips that have been shared with them.
    b. **Role with respect to development project**: End user
    c. **Role with respect to domain**: Potential travel consumer
    d. **Part of organization they work**: Any part
    e. **Impact**: If the friends of Travelers are unhappy with the system, they could impact the potential satisfaction of the Travelers who shared information with them. Additionally, if they are unsatisfied with the sharing features, they are less likely to use the application themselves in the future. By meeting the expectations of these users, we can potentially convert them into Travelers, which will lead to the growth this project requires for our financial goals.

### 1.3.2 Producers

1. Our Customer Service Representatives
    a. **Vision**: A system that is not error-prone and ensures that data is efficiently and correctly sent to external systems. A system that provides them with data about existing reservations and accounts and allows them to easily troubleshoot account problems and make modifications as needed.
    b. **Contribution**: Meet regularly with Development managers to ensure that customer issues labeled Major or higher are being resolved with appropriate SLAs. Provide feedback on those tools and account that they require to access and manipulate data to best serve customers. Regular root-cause analysis on all tickets to feed new features or development work into product requirements
    c. **Impact if their vision is not met**: Travelers are less likely to have their expectations met. Travelers will not see us as a company that is easy to do business with.
2. Software Architects
    a. **Vision**: An elegant design that meets the needs of the business analysts and provides a good coding environment for the software engineers. The architects desire an easily maintainable and modifiable codebase.
    b. **Contribution**: Software architecture for the application and guidance on architecture decisions as the project progresses.

    c. **Impact if their vision is not met**: The Software Engineers will not be as effective in meeting their vision. The software will be difficult to maintain and modify leading to disruption of other business goals and potential high turnover.

3. Software Engineers
    a. **Vision**: Software Engineers expect a system in which the code is easy to understand, modify, and maintain. The tools and stack in place should empower development teams to make effective decisions quickly in line with the architectural vision. It should be easy for Software Engineers to be “good citizens” of the code.
    b. **Contribution**: They design and write the code that defines the application according to the architectural vision.
    c. **Impact if their vision is not met**: The code will become difficult to maintain and manage. Customer issues and desired features are likely to pile up and thus disrupt other business goals. Delivery estimates and timelines will be difficult to make accurately.
4. Quality Assurance Engineers
    a. **Vision**: QA Engineers expect an application that is easy to test with automated tools and which is not prone to unpredictable or difficult to reproduce defects. QA Engineers also expect that unit and integration tests be done by Software Engineers prior to end-to-end automated tests. QA Engineers expect the build, test, and production environments to be as similar as possible, if not sharing resources where risk is not introduced.
    b. **Contribution**: They verify and predict the reliability of the code and report on defects that could impact customer expectations. They also verify when defects have been removed from the system and provide regression tests to ensure that they do not return.
    c. **Impact if their vision is not met**: The application will be challenging to test, leading to more defects and more unexpected defects. Additionally, less automated tests will lead to longer deployment times and less predictable release schedules. More defects will eventually lead to missed customer expectations and unhappy end users.
5. Database Administrators
    a. **Vision**: The DBAs expect data to be easy to maintain and access. They expect that schema or platform changes should not be overly burdensome on overall database maintenance.
    b. **Contribution**: Protect customer data and ensure that data can be accessed by all required services.
    c. **Impact if their vision is not met**: Data presented to users could be unreliable. Data could be lost or unsecured. Any issues with data will lead to end user expectations not being met, as well as disastrous results for the image of our application.
6. Business Analysts

a. **Vision**: Business Analysts expect that the product meets the requirements of end users and the current best practices for making reservations online. They expect that we are making our product marketable and exciting for our customers and partners.
b. **Contribution**: The Business Analysts should meet regularly with the development team to ensure that priorities are in line with business requirements. They should also hold regular information gathering sessions with end users to compare our product with other options in the marketplace.
c. **Impact if their vision is not met**: Our product will not fill a market need and will not succeed. Development work could be wasted on non-critical defects or features. The application code and interface could be overly complicated by not focusing on business-critical features.

## 1.4 Functional Requirements

### 1.4.1 Function Specifications

Overview of implemented functions:

1. Authenticate and Authorize
2. Search for a flight
3. Select a seat for a flight
4. Enter number and types of luggage
5. Select meal for a flight
6. Modify an existing reservation
7. Confirm reservation modifications
8. Select a flight
9. Reserve a flight
10. Sort and filter flight results
11. Review existing reservation details
12. Review all reservations
13. Cancel an existing reservation
14. Register a new user
15. View User Profile
16. Edit User Profile
17. Update User’s Email Address
18. Update User’s Credentials
19. Add a Travel Planner
20. List Travel Planner(s)
21. Remove a Travel Planner
22. Watch a given route
23. List watched routes
24. Remove a watched route

25. Share Itinerary
26. Add Recipient to a Shared Itinerary
27. Pay for a reservation
28. Process travel credits
29. Pay for a reservation modification
30. Add payment details

## Functional Details

1. **<u>Authenticate and Authorize</u>**: The user logs into the system to access their user account. The user is authorized to view and update reservations and profile data for their user account and those trips that have been shared with their account.
   **Input:**
   Username [Traveler]
   Password [Traveler]
   **Output:**
   User is logged into the application
   **Performers:** Travelers, Traveler Planners
   **Resources:** Repository of user accounts, Repository of user credentials, Secure protocol for managing credentials, Method for maintaining identity
   **Performance Locations:** On desktop, phone app, or other browser
   **Precondition:** User must have performed “Register a new user” function so that an account is already present in the system. User requests a login function to trigger the login process. Login can be triggered upon first accessing the page, during trip planning, or during the flight reservation process.
   **Postcondition:** User is logged into the system. User data is then accessible in the application until the user is logged out. User can now reserve and purchase flights associated with their user account.
2. **<u>Search for a flight</u>:** This function is used to search for flights based on date and schedule preferences. On completion, the user is able to see matching flight results and further refine the results. Also, the user can select from the flight results to proceed to the flight reservation process.
   **Input:**
   Date [Traveler]
   Time of travel preferences [Traveler]
   Incoming/Outgoing Airports [Traveler based on internally maintained list of airports]
   Flight type preferences [Traveler]
   Number of travelers [Traveler]
   Child or handicapped travelers [Traveler]
   **Output:**
   Relevant search results [Limit search]
   Selected travel details [Reserve a flight]
   **Performers:** Travelers, Traveler Planners

**Resources:** Existing airport list, calendar database, external flight schedule API
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:** The user has accessed the page. The user can immediately begin customizing the input for the search on the main page of the application. Once the input has been entered, the search can be initiated to display results.
**Postcondition:** Search results are displayed matching the input.

3. **<u>Select a seat for a flight</u>:** If a traveler wants to select a specific seat for a flight, the user can view and select from available seats. Some seats can result in an additional fee to be added to the reservation, such as first class or premium seats.
   **Input:**
   Flight information [Reserve a flight] or [Modify an existing reservation]
   Airplane seating arrangement [External API]
   Available and reserved seats [External API]
   Selected seat [Traveler]
   **Output:** Selected seat [Returned to Reserve a flight or Modify an existing reservation]
   **Performers:** Travelers, Traveler Planners
   **Resources:** External API for seat information, Visualization for seating chart
   **Performance Locations:** On desktop, phone app, or other browser
   **Precondition:**
   **Postcondition:**
4. **<u>Enter number and types of luggage</u>:** Carriers require you to confirm the number of bags that you are carrying on board a flight. Additionally, some carriers charge fees for baggage. Additionally if you have any pets to carry on board, these must also be declared and paid for prior to boarding the flight.
   **Input:**
   Reservation ID [Modify an existing reservation][Reserve a flight]
   Carrier [Modify an existing reservation][Reserve a flight]
   **Output:**
   Calculated cost of the luggage for the selected flight
   The selected luggage for the flight
   **Performers:** Traveler, Traveler Planners
   **Resources:** Luggage cost structure by carrier
   **Performance Locations:** On desktop, phone app, or other browser
   **Precondition:** User has either initiated the process to reserve a flight or has requested to modify an existing reservation and add luggage.
   **Postcondition:** The selected baggage cost and baggage information is sent to the next step of the reservation or reservation modification process.
5. **<u>Select meal for a flight</u>:** On some flights, users can select from several predetermined meal options. Some of these meal options have an associated fee.
   **Input:**
   Reservation ID [Modify an existing reservation][Reserve a flight]
   Carrier [Modify an existing reservation][Reserve a flight]
   Meal options on the selected flight [External API]

**Output:**
Calculated cost of the meal for the selected flight
The selected meal for the flight
**Performers:** Traveler, Traveler Planners
**Resources:** Meals available on the selected flights with associated costs
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:** User has either initiated the process to reserve a flight or has requested to modify an existing reservation and add/select a meal.
**Postcondition:** The selected meal cost and meal selection is sent to the next step of the reservation or reservation modification process.

6. **<u>Modify an existing reservation</u>:** An authenticated user can update an existing reservation by adding a seat, baggage, or dietary selection, or by rescheduling the entire trip. All such modifications can result in an additional charge that is collected after the modification is confirmed.
   **Input:**
   Type of change to make (bag, seat, meal, schedule) [Traveler]
   Reservation to change [Review existing reservation details]
   **Output:** Type of change to make and which reservation to modify
   **Performers:** Traveler, Traveler Planners
   **Resources:** Traveler data, Reservation data
   **Performance Locations:** On desktop, phone app, or other browser
   **Precondition:** An authenticated user has viewed an existing reservation and selected to update the reservation.
   **Postcondition:** The user is transferred to the next correct process for updated the reservation related to the requested type of modification.
7. **<u>Confirm reservation modifications</u>:** After a user has selected to modify a reservation and then gone through the process of selecting the modification, the user must confirm the modification prior to paying for the changes.
   **Input:**
   Type of change to make [Modify an existing reservation]
   Reservation to change [Modify an existing reservation]
   Seat selection [Select a seat for a flight]
   Meal selection [Select meal for a flight]
   Baggage selection [Enter number and types of luggage]
   **Output:** Total cost and Fee/Fare Breakdown [Sent to *Pay for a reservation modification*]
   **Performers:** Traveler, Traveler Planners
   **Resources:** Reservation data, Fee and Fare data, External reservation API to confirm updates
   **Performance Locations:** On desktop, phone app, or other browser
   **Precondition:** An authenticated user has viewed an existing reservation and selected to update the reservation. The user has then selected the specific details of what to change, such as schedule or selecting a seat for the flight.

**Postcondition:** The correct total cost of the modification has been sent to the payment processor and the modification has been confirmed with the external reservation API.

8. **<u>Select a flight</u>:** After a user has searched for a flight and received flight options, the user can choose to select a single flight option to proceed with the reservation process.
   **Input:**
   Flight option [Traveler]
   **Output:**
   Access to reserve flight option
   **Performers:** Travelers, Traveler Planners
   **Resources:** Same as those required by Search for a flight.
   **Performance Locations:** On desktop, phone app, or other browser
   **Precondition:** User has searched for a flight and received search results.
   **Postcondition:** User has access to reserve the selected flight option.
9. **<u>Reserve a flight</u>:** User has searched for and selected a flight and now wants to reserve that flight to ensure that they have a ticket for the flight. A reservation involves some form of payment and returns the confirmed reservation in return. Much of the data for this function can be accessed from the traveler's account if they have performed Authenticate and Authorize, otherwise they must enter the data during reservation.
   **Input:**
   Flight option [Select a flight]
   Baggage [Enter number and types of luggage]
   Number of tickets [Traveler]
   Traveler name [Traveler][Authenticate and Authorize]
   Traveler date of birth [Traveler][Authenticate and Authorize]
   Street Address [Traveler][Authenticate and Authorize]
   Gender [Traveler][Authenticate and Authorize]
   Dietary Preference [Select meal for a flight]
   Seat preference [Traveler][Authenticate and Authorize] - used to request a seat in the event that the user does not select a seat
   Seat [Select a seat] (*Optional*)
   Travel Reward Programs [Traveler][Authenticate and Authorize] (*Optional*)
   Known Traveler Number [Traveler][Authenticate and Authorize] (*Optional*)
   **Output:**
   Reservation confirmation on screen and via email
   Reservation saved to user account if they logged in prior to reservation
   If reservation fails, the user is notified of the failure
   Total cost and Fee/Fare Breakdown [Sent to Pay for a reservation]
   **Performers:**
   Traveler, Traveler Planners
   **Resources:** Traveler database, External reservation API, External fee/fares calculation API, Calendar component to verify birthdates, List of dietary options for airlines, Travel reward number verification system, Known Traveler Number verification system
   **Performance Locations:** On desktop, phone app, or other browser

**Precondition:**
User has searched for and selected a flight
User has optionally performed Authenticate and Authorize to access their travel information and preferences
User has selected to reserve a selected flight option
**Postcondition:**
User has a confirmed reservation on the flight or a notice of failure for the selected reservation

10. **Sort and filter flight results:** A user has searched for flights that match their search criteria and then wants to sort and filter those results, prior to making a selection. The search results should be displayed and sortable/filterable based on travel duration, time of day, price, and carrier.
    **Input:**
    Travel duration [Search for a flight]
    Schedule [Search for a flight]
    Price [Search for a flight]
    Carriers [Search for a flight]
    Sort order [Traveler]
    Sort field [Traveler]
    Filter field [Traveler]
    **Output:**
    Filtered and sorted list of results
    **Performers:** Travelers, Traveler Planners
    **Resources:** Sorting and filtering capabilities
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:**
    User has performed Search for a flight and received search results
    **Postcondition:**
    User has search results that are sorted and filtered according to their specifications
11. **Review existing reservation details:** A user can select an existing reservation from the list of all reservations for the account or a shared account (in the case of a Planner) to view details for the account. The view should display payment history, already selected seats, type of aircraft, schedule, and other relevant details.
    **Input:**
    Selected reservation [Review all reservations]
    **Output:**
    The following information is displayed to the user: flight number, flight duration, flight schedule, traveler details, reserved seats, reserved baggage, dietary selections, rewards program information, transaction history
    **Performers:** Travelers, Traveler Planners
    **Resources:** Traveler data, Reservation data, Transaction history
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:** User has logged into the application and viewed all reservations for the

account. User has selected to view the details for a single reservation.
**Postcondition:** User is presented with all details for the selected reservation.

12. **<u>Review all reservations</u>**: An authenticated user can review a list of all existing reservations made or assigned to their account. These reservations can be reservations they themselves made, a Traveler Planner made for them, or which were shared with their account by a fellow traveler. An empty list should also display with appropriate text in the event that a user account has no existing reservations. The reservation list should be able to display both past and future reservations as specified by the user.
    **Input:**
    User Account [Authenticate and Authorize]
    Existing reservations [Traveler and reservation data]
    **Output:**
    A list of reservations sorted by travel date and filterable by past and future
    **Performers:** Travelers, Traveler Planners
    **Resources:** Traveler data, Reservation data with historic data and a relationship to Traveler data.
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:** Traveler has created an account and logged in. User has requested to see a list of all existing reservations.
    **Postcondition:** User is shown the list of all reservations. Traveler Planners can also see a list of reservations for a selected traveler for whom they have been given travel authority. Travelers who have made reservations as a group can see details for reservations for other travelers that are part of the same reservation.
13. **<u>Cancel an existing reservation</u>:** If an authenticated user no longer wants to keep an existing reservation, the user can cancel the reservation directly through the application. Canceling a reservation generally results in some form of travel credit or returned payment for the user. However, a calculation must be performed to determine how much credit or currency is appropriate for the cancellation based on the time till departure and the type of flight. The calculation also includes how that refund can be issued (some refunds required travel credits only and not a refund of currency).
    **Input:**
    Selected reservation [Review existing reservation details]
    **Output:**
    Cancellation sent to external reservation API
    Correct refund calculation sent to *Process travel credits* function.
    **Performers:** Travelers, Traveler Planners
    **Resources:** Information required to calculate appropriate refund, External reservation API
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:** User has viewed the details for an existing reservation and requested to cancel the reservation. The user has confirmed the cancellation.
    **Postcondition:** The reservation is cancelled both with the external reservation API and internally. The correct refund calculation is sent to the *Process travel credits* function.

14. **Register a new user:** A new user requests to create a new account to store their travel preferences and eventually their associated reservations and methods of payment. As part of registering, a user must enter basic biographical information before creating the account. The user can also enter some preferences that affect how their reservations are made by default, such as dietary preferences and seat preferences.
**Input:**
Username [Traveler]
Password [Traveler]
Full name [Traveler]
Email Address [Traveler]
Street Address [Traveler]
Gender [Traveler]
Birthday [Traveler]
Dietary Preference [Traveler] (*Optional*)
Seat Preference [Traveler] (*Optional*)
Travel Reward Programs [Traveler] (*Optional*)
Known Traveler Number [Traveler] (*Optional*)
**Output:**
Traveler ID added to database
Email confirmation
**Performers:**
Traveler, Traveler Planner
**Resources:**
Traveler database, Calendar component to verify birthdates, List of dietary options for airlines, Travel reward number verification system, Known Traveler Number verification system
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:**
User has requested to register a new user account
**Postcondition:**
User is registered in the system and can login to view and update account details. User can also reserve flight reservations for their account.

15. **View User Profile:** A user can use this function to view their profile information as registered in the application. Traveler Planner will only have read access
**Input:**
Traveler ID [Authenticate and Authorize]
**Output:**
User's Profile as stored in the application. It will include any of the following if available: Username, Password (Hashed), Full name, Email Address, Street Address, Gender, Birthday, Dietary Preference, Seat Preference, Travel Reward Programs, TSA Known Traveler Number, etc. but not be editable in this display
**Performers:**
Traveler, Traveler Planner

**Resources:**
Traveler database, Calendar component to verify birthdates, List of dietary options for airlines, Travel reward number verification system, Known Traveler Number verification system
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:**
User is registered in the system and has logged in.  User has requested the View User profile function.
ActionType: View
**Postcondition:**
The user’s full profile details as stored in the system is presented to the user.

16. **Edit User Profile:** A user can use this function to view their profile information as registered in the application.  Traveler Planner will only have read access
    **Input:**
    Traveler ID [View User Profile]
    Full name [Traveler]
    Street Address [Traveler]
    Gender [Traveler]
    Birthday [Traveler]
    Dietary Preference [Traveler] (*Optional*)
    Seat Preference [Traveler] (*Optional*)
    Travel Reward Programs [Traveler] (*Optional*)
    Known Traveler Number [Traveler] (*Optional*)
    Payment information [Enter Payment data]
    **Output:**
    User’s Profile as stored in the application is updated with the new input values provided.
    **Performers:**
    Traveler
    **Resources:**
    Traveler database, Calendar component to verify birthdates, List of dietary options for airlines, Travel reward number verification system, Known Traveler Number verification system
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:**
    User is registered in the system and has logged in.  User has requested the Edit User profile function.
    ActionType: Edit
    **Postcondition:**
    The user’s profile details is updated with the new input values.
17. **Update User’s Email Address:** A user can use this function to update their email address registered in the system.  A validation email is sent to the new address with a 10 minute timeout for user to respond by clicking on the validation link.  If the user validates the new email address, the transaction proceeds and the email address is updated.

**Input:**
Traveler ID [View User Profile]
New Email [Traveler]
**Output:**
A new email address verification email is sent to the specified email address and the user is asked to confirm the address within 10 minutes or the request will timeout. If new email is validated, an email address change notification is sent to both the old and new email address and the User's Profile is updated with the new email address.
**Performers:**
Traveler
**Resources:**
Traveler database,
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:**
User is registered in the system and has logged in. User has requested the Edit Email Address function.
ActionType: Update
**Postcondition:**
If new email is validated, email change notification is sent and User profile is updated with new email address.

18. **Update User's Credentials:** A user can use this function to update their username and/or password in the system. A confirmation email is sent to the user's email address after the user's credentials are updated..
**Input:**
Traveler ID [View User Profile]
New Username [Traveler] (optional)
New Password [Traveler] (optional)
**Output:**
If a new username or password is specified and meets complexity requirements, update the credentials in the user's profile and send a confirmation email to the user's email address on record.
**Performers:**
Traveler
**Resources:**
Traveler database
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:**
User is registered in the system and has logged in. User has requested the Update Credentials function.
ActionType: Update
**Postcondition:**
If a compliant new username or new password is specified, update credentials in user's profile and send confirmation email to address on record.

19. **<u>Add a Travel Planner</u>:** This function allows the user to designate another person as a travel planner.  The planner will have some access to the user's profile and is empowered to plan trips or modify existing reservations on behalf of the user.  Planners are typically administrative assistants or travel agents.
    **Input:**
    Traveler ID [Authenticate and Authorize]
    Planner Full Name [Traveler]
    Planner Email Address [ Traveler]
    **Output:**
    Traveler's Authorized Planner List [System]
    **Performers:** Traveler
    **Resources:** Traveler Database (Traveler ID), Traveler's Authorized Planner List
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:**
    User is registered in the system and has logged in.  User has requested the Assign a Travel Planner function function.
    ActionType: Add
    **Postcondition:** Traveler's Authorized Planner List is produced
20. **<u>List Travel Planner(s)</u>:**  This function is used by a traveler to return the traveler's Authorized Planner list.  Planners are typically administrative assistants or travel agents.
    **Input:**
    Traveler ID [Authenticate and Authorize]
    **Output:**
    Traveler's Authorized Planner List [System]
    **Performers:** Traveler
    **Resources:** Traveler Database (Traveler ID), Traveler's Authorized Planner List
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:**
    User is registered in the system and has logged in.  User has requested List Travel Planner function
    ActionType: List
    **Postcondition:**
    Traveler's Authorized Planner List is produced
21. **<u>Remove a Travel Planner</u>:** This function allows the user to remove a user's existing travel planner.  The planner who is removed will lose access to the user's profile and can no longer plan trips or modify existing reservations on behalf of the user.  Planners are typically administrative assistants or travel agents.
    **Input:**
    Traveler ID [Authenticate and Authorize]
    Planner ID [List Travel Planner]
    **Output:**
    Traveler's Authorized Planner List [System]
    **Performers:** Travelers

**Resources:** Traveler Database (Traveler ID), Traveler's Authorized Planner List
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:**
User is registered in the system and has logged in. User has requested Remove a Travel Planner function.
ActionType: Remove
**Postcondition:**
Traveler's Authorized Planner List is updated

22. **Watch a given route:** This function is used by a traveler to register a low fare watch for a given route a specified duration. An optional max fare the traveler is willing to pay can be specified otherwise the system uses an internal 30 day average to determine max fare. On completion, the function sends a Route Watch confirmation email to the traveler and the traveler's route watch list in the system is updated with the new Route Watch.
    **Input:**
    Traveler ID [Authenticate and Authorize]
    Departure City [User]
    Arrival City [User]
    Watch Period [User]
    Max Fare [User] (Optional)
    Alert Type [User]
    **Output:**
    Email [External]
    Traveler's Route Watch List Updated [System]
    **Performers:** Traveler
    **Resources:** Existing airport list, Traveler Database (Traveler ID), Average Fares, System Task Runner, Current Fares
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:**
    User is registered in the system and has logged in. User has requested Watch a Given Route function.
    ActionType: Add
    **Postcondition:**
    Route Watch confirmation email sent to traveler and Traveler Route Watch List updated
23. **List watched routes:** This function is used by a traveler to return the traveler's Route Watch list.
    **Input:**
    Traveler ID [Authenticate and Authorize]
    **Output:**
    Traveler's Route Watch List [System]
    **Performers:** Traveler
    **Resources:** Traveler Database (Traveler ID), Traveler Route Watch List

**Performance Locations:** On desktop, phone app, or other browser
**Precondition:**
User is registered in the system and has logged in.  User has requested List Watched Routes function
ActionType: List
**Postcondition:**
Traveler Route Watch List is produced

24. **<u>Remove a watched route</u>:** This function is used by a traveler to delete a registered low fare watch for a given route.  On completion, the function sends a Watched Route Removal confirmation email to the traveler and removes the specified traveler's route watch from the traveler's route watch list.
**Input:**
RouteWatch ID [List Watched Routes]
Action Type (List) [Traveler]
Action Type (Remove) [Traveler]
**Output:**
Email [External]
Traveler's Route Watch List updated [System]
**Performers:** Traveler
**Resources:** Traveler Database (Traveler ID), Traveler Route Watch List
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:**
User is registered in the system and has logged in.  User has requested Unwatch a watched route function
ActionType: Remove
**Postcondition:**
Route Watch Removal confirmation email sent to traveler and Traveler Route Watch List is updated

25. **<u>Share Itinerary</u>:** This function is used by a traveler to mark an itinerary to be publically shared via email.
**Input:**
Traveler ID [Authenticate and Authorize]
Reservation ID [Traveler and reservation data]
**Output:**
Shared Itinerary ID [Add Recipient to a Shared Itinerary]
**Performers:** Traveler
**Resources:** Traveler Database (Traveler ID), Traveler's Shared Itinerary List
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:**
User is registered in the system and has logged in.  User has Planned a Trip. User requested Share Itinerary function.
ActionType: Share
**Postcondition:**

User is sent to the Add Recipient to Shared Itinerary Function.

26. **<u>Add Recipient to a Shared Itinerary</u>:** This function is used by a traveler to add a new recipient to a Shared Itinerary.  On completion, the function sends “An Itinerary is Shared with you” email notification to the recipient the itinerary is shared with.  At this time, we will not implement social sharing but it could be a feature in future.
    **Input:**
    Traveler ID [Authenticate and Authorize]
    Shared Itinerary ID [Share Itinerary] or Shared Itinerary ID [List Shared Itineraries]
    Shared Recipient Full Name [Traveler]
    Shared Recipient Email Address [ Traveler]
    **Output:**
    Traveler’s Shared Itinerary List [System]
    **Performers:** Traveler
    **Resources:** Traveler Database (Traveler ID)
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:**
    User is registered in the system and has logged in.  User has Planned a Trip and created a travel reservation. User requested Add Recipient to Shared Itinerary function.
    ActionType: Add
    **Postcondition:**
    “An Itinerary is Shared with you” email is sent to recipient.
27. **<u>Pay for a reservation</u>:** After selecting a flight and beginning the Reserve a flight function, the user must pay for the travel plans prior to confirming the reservation. This payment should be in a format that can be immediately collected at the time of reservation, such as credit card.
    **Input:**
    Total cost with fee/fare breakdown [Reserve a flight]
    Payment account information [Add payment details] [Authenticate and Authorize]
    **Output:**
    Payment approval for Reserve a flight function.
    **Performers:**
    Travelers, Traveler Planners
    **Resources:**
    Credit card payment service to transmit payment, Traveler database with secure and encrypted payment retrieval
    **Performance Locations:** On desktop, phone app, or other browser
    **Precondition:** User has started the Reserve a flight function and has proceeded to the payment entry form. If the user wants to use stored payment data, they can log in to access their stored payment account.
    **Postcondition:** User payment has been collected and the reservation can be confirmed.
28. **<u>Process travel credits</u>:** After an authenticated user has cancelled an existing reservation, some travel credits or payment will be returned to them based on the type of cancellation. The appropriate amount to be refunded and how that refund can be issued

are calculated as part of the Cancel an existing reservation function
**Input:**
Refund Amount [Cancel an existing reservation]
Types of Allowable Refund [Cancel an existing reservation]
Desired refund type [Traveler]
**Output:**
Refund is processed as currency or travel credits available on the user's account. If a cancellation is made for another traveler, the credit must be attributed to the original payment form or travel credits associated with the travel account that originally paid for the travel.
**Performers:** Traveler, Traveler Planners
**Resources:** Travel credit database, Traveler data, Payment data
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:** User has performed the *Cancel an existing reservation* function
**Postcondition:** Refund is processed and funds or credits transferred.

29. **<u>Pay for a reservation modification</u>:** Users can modify an existing reservation in several different ways. After selecting a new flight, updating seats, adding baggage, or meal plan, a user must pay for the modification before the change is confirmed and saved to their reservation. This payment should be in a format that can be immediately collected at the time of the modification, such as credit card.
**Input:**
Total cost with fee/fare breakdown [Modify an existing reservation]
Payment information [Add payment details] [Authenticate and Authorize]
**Output:**
Payment approval for "Modify an existing reservation function."
**Performers:** Travelers, Traveler Planners
**Resources:** Credit card payment service to transmit payment, Traveler database with secure and encrypted payment retrieval
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:** User has started the Modify an existing reservation function and has proceeded to the payment entry form. If the user wants to use stored payment data, they can log in to access their stored payment account.
**Postcondition:** User payment has been collected and the reservation modification can be confirmed.

30. **<u>Add payment details</u>:** Travelers can add a payment method to use for purchasing reservations. Authenticated users can optionally store this data for future transactions when logged into their account.
**Input:** Credit card information [Traveler]
**Output:** Payment information is returned to user account for storage by "Edit user account details" or transmitted to "Pay for a reservation modification" or to "Pay for a reservation."
**Performers:** Travelers, Traveler Planners
**Resources:** Credit card payment service to verify payment details, Traveler database

with ability to securely store encrypted payment details
**Performance Locations:** On desktop, phone app, or other browser
**Precondition:** User has requested to pay for a reservation, pay for a reservation modification, or store a new method of payment on their user account.
**Postcondition:** Payment details are available for the current transaction or for storage on the user account.

### 1.4.2 Scenario Specifications

1. **Plan a trip:**
   **Sequence of Functions:**
   *Function1*: Search for flights
   *Function3*: Sort and filter flight results *Optional*)
   *Function4*: Select a flight
   *Function5*: Register a new user (*Optional*)
   *Function6*: Authenticate and Authorize (*Optional*)
   *Function7*: Reserve a flight
   *Function8*: Add payment details
   *Function9*: Pay for a reservation
   **Environment**: User wants to search for and reserve a ticket for air travel. User wants to compare multiple possibilities for the trip and optionally filter the results. The user registers a new account or logs in with an existing account to maintain the reservation on their account.
   **Preconditions**: User has accessed the site
   **Postconditions**:
   If registering a new user, user account is created and has access to the new reservation
   User has a saved reservation, visible in the reservation list
   User can view the details for the reservation
   Reservation is transferred to external carrier system
   All required payments for this reservation should be processed.

2. **Review an existing trip:**
   **Sequence of Functions:**
   *Function1*: Authenticate and Authorize
   *Function2*: Review all reservations
   *Function3*: *Review existing reservation details*
   **Environment**: User wants to view the details for a specific trip that has already been reserved in the system. They want to see all relevant details for the trip in one location.
   **Preconditions**:
   User has performed Register a new user
   User has performed Reserve a flight
   User has performed Pay for a reservation
   **Postconditions**:

Details for the reservation are displayed to the user

3. **<u>Cancel an existing trip:</u>**
   **Sequence of Functions:**
   *Function1*: Cancel an existing reservation
   *Function2*: Process travel credits
   **Environment**: User wants to cancel a trip that has been reserved through the system. They want any credits for the trip to be returned to their payment account or maintained on the site for future travel.
   **Preconditions**:
   User has performed Review an existing trip
   **Postconditions**:
   The reservation is marked as cancelled on the site
   Updates are transferred to external carrier system
   Travel credits are either saved to the site or paid back to saved payment method

4. **<u>Update Trip Details</u>**:
   **Sequence of Functions:**
   *Function1*: Modify an existing reservation
   *Function2*: Enter number and types of luggage (*Optional*)
   *Function3*: Select meal for a flight (*Optional*)
   *Function4*: Select a seat for a flight (*Optional*)
   *Function5*: Confirm reservation modifications
   *Function6*: Pay for a reservation modification
   **Environment**: User decides to modify or select details for an existing reservation. For example, the user might want to add luggage requirements or select or change a seat for the flight.
   **Preconditions**:
   User has performed Review an existing trip
   **Postconditions**:
   User has a modified reservation
   Updates are transferred to external carrier system
   All required payments for this change should be processed.

5. **<u>Reschedule an existing trip</u>**:
   **Sequence of Functions:**
   *Function1*: Modify an existing reservation
   *Function2*: Search for a flight
   *Function3*: Select a flight
   *Function4*: Confirm reservation modifications
   *Function5*: Pay for a reservation modification
   **Environment**: User decides to update an existing reservation to select a completely different flight reservation. The existing reservation will be cancelled and replaced by the

new reservation. Any credits for this reservation should be applied to the new ticket. Any penalties, fines, or difference in fare should be charged at the time of the reschedule.
**Preconditions**:
User has performed Review an existing trip
**Postconditions**:
Reservation is updated to new schedule
Reservation is booked in external carrier system
All required payments for this change should be processed.

6. **Assign and Unassign a Travel Planner**:
   **Sequence of Functions:**
   *Function1*: Assign a Travel Planner
   *Function2*: List Travel Planner
   *Function3*: Remove a Travel Planner
   **Environment**: A registered user (traveler) wants another person to make travel plans on behalf of the registered user. The performing user (planner or assignee) is not traveling as part of this reservation. The planner might be a administrative assistant or a travel agent planning trips for a registered user.
   **Preconditions**:
   User (Traveler) has performed Register a new user
   **Postconditions**:
   Email is sent notifying Assignee of their role as Planner to the User.
   Email contains a link to the User's reservation list page on the site.
   Planner (Assignee) is required to perform Register a new user function if not registered in the application.
   Planner has modify access to the user's (assignor's) reservation list and user travel preferences
   Assigned planner has read access to remaining portions of the user's profile

7. **Share a trip Itinerary (overview)**:
   **Sequence of Functions:**
   *Function1*: Share Itinerary
   *Function2*: Add Recipient to a Shared Itinerary
   **Environment**: User has booked a trip and wants to share only the itinerary (not e-Ticket or payment details etc.) publically with a specified person (say a friend) who is not a traveler on the trip.
   **Preconditions**:
   User has performed Review an existing trip or Plan a trip
   **Postconditions**:
   Email notification is sent to recipient with a link to the shared itinerary.  Recipient can click on link to view shared itinerary in a browser
   Recipient does not have to login or register to view the itinerary

8. **<u>Update Profile and Preferences</u>**:
   **Sequence of Functions:**
   *Function1:* View User profile
   *Function2*: Edit User profile
   *Function3*: Update User's Email Address *(optional)*
   *Function3*: Update User's Credentials *(optional)*
   **Environment**: User wants to update their User profile, Email address on record, or their user credentials (username and/or password)
   **Preconditions**:
   User is registered in system and is logged in.  User wants to view their profile or edit profile, email, or credentials
   **Postconditions**:
   User's profile is updated as specified after validation.  Confirmation email of the update performed is sent to the email address (and new email address if specified) on record.

10. **<u>Add a Fare Watch Alert</u>**:
    **Sequence of Functions:**
    *Function1*: Authenticate and Authorize
    *Function2*: Watch a given Route
    Function3: List Watched Routes
    **Environment**: User wants to register a low airfare watch for a given travel route and look forward a specified number of days upto one year.
    **Preconditions**:
    User (Traveler) has performed Register a new user and is logged in.  User has requested Add a Fare Watch Alert function.
    **Postconditions**:
    Traveler's Route Watch List is updated

11. **<u>Remove a Fare Watch Alert</u>**:
    **Sequence of Functions:**
    *Function1*: Authenticate and Authorize
    *Function2*: List Watched Routes
    Function3: Remove a Watched Route
    **Environment**: User wants to remove a registered low airfare watch for a given travel route.
    **Preconditions**:
    User (Traveler) has performed Register a new user and is logged in. User has at least one Fare Watch Alerts registered.  User has requested Remove a Fare Watch Alert function
    **Postconditions**:
    Traveler's Route Watch List is updated

### 1.4.3 Essential Scenario

The essential scenario would be **“Plan a trip”** because this is the main purpose for the site. If users can not plan a trip, then none of the other scenarios or functions on this site have a purpose. Additionally, all revenue for the site is generated by users paying for reservations through the site. This is also the process that must be the easiest to perform and understand to ensure a competitive edge in the marketplace. Special focus on this scenario will ensure that the site provides value to our users, is profitable, and has competitive advantage.

## 1.5. Qualities and Constraints

1. **Usability:**
   **Description:** The user should be able to navigate and interact with the system with minimal training. The interface should be self-explanatory, easy to use, and intuitive.
   **Category:** Usability
   **Stakeholder Source:** Travelers, Traveler Planners, Business Analysts, QA Engineers, Customer Service Representatives
   **Scope:** The scope of this quality applies to all functions (and parts of the system) that require end-user interaction
   **Evaluation:** UI designers will provide consistent styling and interface components and perform regular usability testing using A/B view methodology. QA Engineers will test the application similar to an end-user and ascertain whether all the specifications from the requirements document are covered or not.

2. **Availability:**
   **Description:** This system is expected to be available 24x7 with a four 9's (99.99%) uptime goal. The system will be accessed by the end-users from any internet accessible location across different timezones. If the system is down, then it will impact the reputation of the site and the end-user's ability to schedule and plan trips.
   **Category**: Availability
   **Stakeholder Source:** Travelers, Traveler Planners, Business Analysts, QA Engineers, Customer Service Representatives
   **Scope:** The scope of this quality applies to all functions (and parts of the system) that require end-user interaction
   **Evaluation:** Based on an SLA of 24x7 and a four 9's uptime goal, the system should not be unavailable for more than 53 minutes in a calendar year. These SLAs can be guaranteed via all major public cloud services.

3. **Performance:**
   **Description:** The user should not have to wait for extended periods of time at any point when using the system.
   **Category**: Performance

**Stakeholder Source:** Traveler Planners, Business Analysts, QA Engineers, Customer Service Representatives, Database Administrators, Software Architect
**Scope:** Entire Application
**Evaluation:** Measure the time it takes for various functions to complete

4. **Security:**
   **Description:** Authenticated access to the system should be secure and user credentials should not be compromised. SSL can be used to secure the transmission of sensitive information. User identifying (personal) information should be encrypted when stored in the database.  Collection, transmission, and storage of payment (CREDIT CARD) information or cardholder data must comply with the Payment Card Industry Data Security Standard (PCI DSS).
   **Category**: Security
   **Stakeholder Source:** Travelers, Traveler Planners, Business Analysts, QA Engineers, Customer Service Representatives, Database Administrators
   **Scope:** Entire Application (Authentication in the Login module, Payment information in the payment module)
   **Evaluation:** Use internal and external vulnerability scans using approved PCI Security Standard Council (SSC) Vendors

5. **Scalability:**
   **Description:** The application should be able to handle large numbers of users and have ability to increase capacity on demand in response to growth or during peak travel seasons.
   **Category**: Performance
   **Stakeholder Source:** Travelers, Traveler Planners, Business Analysts, QA Engineers, Customer Service Representatives
   **Scope:** Entire Application
   **Evaluation:** Can use automated load (transaction) generators to test system's elasticity to handle simulated increased load.

6. **Extensibility:**
   **Description:** The application should be extensible. Tight coupling between modules should be avoided in order to facilitate the addition of new features or capabilities with minimal design effort.
   **Category**: Extensibility
   **Stakeholder Source:** Software Engineers, Software Architects, Database Administrators
   **Scope:** Entire Application
   **Evaluation:** Review system design and ensure coding best practices are followed with the use of appropriate design pattern where applicable.

7. **Data backup and recovery:**

**Description:** All critical user and system data must be completely (FULL Backups) backed up on an established daily or weekly schedule. In addition, continuous incremental backups must occur to allow point-in-time restores when needed.
**Category**: Data Recovery
**Stakeholder Source:** Database Administrators, Travelers, Traveler Planners, Business Analysts, QA Engineers, Customer Service Representatives
**Scope:** Entire Application
**Evaluation:** Perform scheduled (monthly or quarterly) recovery into development systems to validate backup and restore capability.

8. **Maintainability:**
**Description:** The application should be created using proper coding principles and design patterns with controlled releases from dev to test to production. It should easily facilitate controlled roll-outs of incremental updates and bug fixes.
**Category**: Maintainability
**Stakeholder Source:** Software Engineers, Software Architects, QA Engineers, Customer Service Representatives
**Scope:** Entire Application
**Evaluation:** Perform scheduled code reviews at each milestone. Deployment and regression test times should be regularly evaluated for inappropriate growth.

9. **Project Cost:**
**Description:** The application should be created within the specified budget without any overrun. The budget covers the final cost of for deploying the application and includes any necessary hardware, infrastructure and software costs.
**Category**: Project constraint
**Stakeholder Source:** Business Analysts
**Scope:** Entire Application
**Evaluation:** Milestones should be set for the project. At each milestone, costs incurred must be evaluated to ensure no overrun.

10. **Project Schedule:**
**Description:** At the inception of the project, the project manager (PM) will create a tentative schedule based on all stakeholder inputs and inclusive of necessary schedule buffers. The application must be delivered within the proposed (scheduled) timeline. The time-to-deliver must not be overrun. Track the project at the regular intervals to identify risk and develop a mitigation plan.
**Category**: Project constraint
**Stakeholder Source:** Business Analysts
**Scope:** Entire Application
**Evaluation:**. Milestones should be set for the project. At each milestone, deadlines and tasks completed must be evaluated for the delivered product.

## 1.6. Deployment Environment

**Server & Computing platforms:** Managed Hosting Provider or Public Cloud Provider
**Description:** We do not have an in-house data center and should not procure server hardware in-house. This solution will need to be deployed hardware using a managed hosting provider offering or a public cloud computing provider using a suitable mix of Infrastructure as a Service (IAAS), Platform as a Service (PAAS), or Software as a Service (SAAS) components. There are no minimum hardware specifications in terms of processor type, processor speed, system memory, storage capacity, or network connectivity so long as the deployed solution meets the specified application requirements.
**Stakeholder Source:** Software Architect

**Solution Architecture:** 3-Tier
**Description:** This application must be developed using a multi-tier (n-tier) architecture where the presentation, application processing, and data management functions are physically (or logically) separated. Either the Java J2EE framework or Microsoft .Net Framework may be utilized.

> Presentation Layer: This is also called the client tier and is the topmost level of the application. This tier is presented to the end user and contains the user interface. The purpose of this layer is to translate the application function inputs and outputs into something the user can understand. The preference is to use a web browser with HTML5 as the presentation layer so that it can work on any end user device regardless of type.
> Application (Logic) Layer: Also called the middle tier, this tier contains the application's business logic and performs all necessary computation. The encapsulated business objects and their functions will reside in this tier and it moves and processes data between the two surrounding layers.
> Data Layer: This tier consists of the database servers. Information is stored and retrieved from this tier. Data must be kept separate from the application logic to improve scalability, portability, and extensibility. Information is passed to the logic tier for processing and then eventually back to the user in a consumable format.

**Stakeholder Source:** Software Architect

**Operating System:** Any Linux distribution or Windows Server 2012 R2
**Description:** Since we will use a managed hosting provider or a cloud provider, we are flexible on what type of operating system to use. Linux is predominant in most cloud deployments due to cost and perceived as more secure. But if the selected hosting or cloud provider has a preference for Windows that should not be excluded. Total Cost of ownership and ease of implementation should be the predominant deciding factors.
**Stakeholder Source:** Software Architect, CEO

**Programming Platform:** Java or .Net (C#)

**Description:** Based on our solution architecture, we can develop this solution using Java or a suitable .Net language. Our developers are proficient in Java and C#.  We want to take advantage of each environment's built-in features that support portability, encapsulation, and easy extensibility.  Code development must use object-oriented programming concepts and accepted design patterns.
**Stakeholder Source:** Software Architect, Software Engineers

**Database:** MySQL, SQL Server, PostgreSQL Oracle, or Other
**Description:** Any relational database platform that supports CRUD (Create, Retrieve, Update, Delete) operations with transaction support (commit, rollback) is suitable.  The selected platform must support dynamic scalability, provide good performance and high availability, and align with the selected hosting provider's optimized offerings. Licensing cost is an important factor to consider as well.
**Stakeholder Source:** DBA, Software Architect

**Application Server:** Any (Open Source) suitable for Java or .Net
**Description:** Use a suitable (even open source) application server that supports either J2EE or the .Net framework and aligns with the solution architecture stack for the middleware application layer.  A solution with a lightweight, modular design, and fast startup and ability to load classes on demand is a plus.  Suitable PAAS or SAAS cloud offerings can be considered if meets the requirement.
**Stakeholder Source:** Software Architect

**Third-Party Software:**
**Description:** Appropriate third-party software that aligns with the solution architecture can be considered if it provides any necessary functionality that would otherwise have to be built. Considerations include modules that provide SSO or other integration functions.
**Stakeholder Source:** Software Architect