EECS 3550-001: Software Engineering

Air 3550

Group 11

Requirements Document

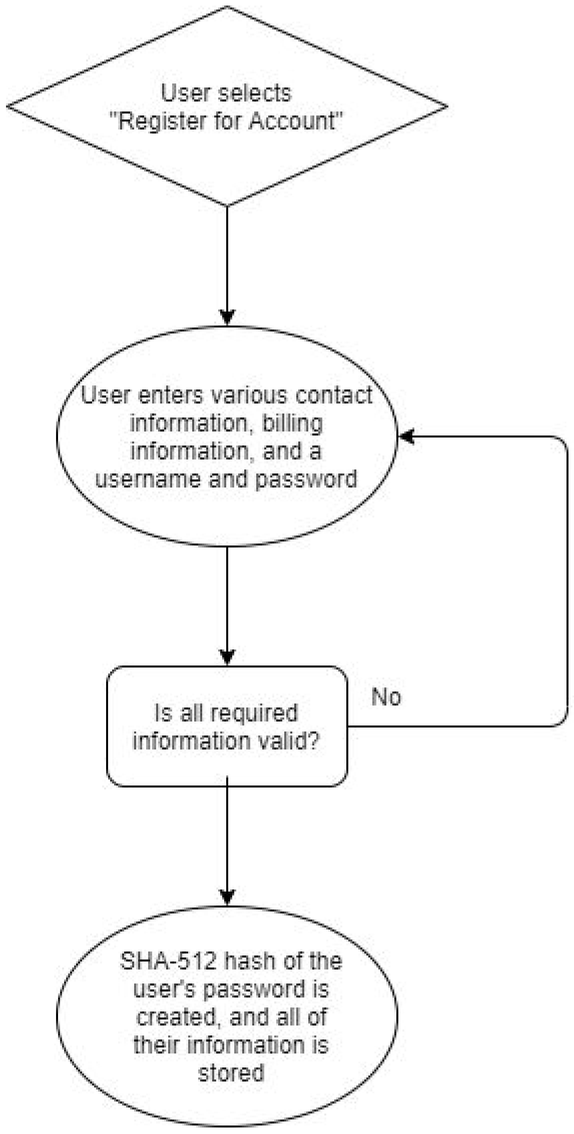
02/28/2021

James Golden, Edward Walsh, & Quinn Kleinfelter

Use Case: Register for Account
Primary actor: Someone who wishes to have an account
Goal in context: To allow people to register for an account in the flight booking system
Trigger: User selects the "Register" button on the Air-3550 home page
Scenario:

1. User is on the Air-3550 home page
2. User clicks "Register" and is redirected to a form where their information is recorded

Exceptions:
Priority: Highest Priority, if a user cannot register for an account they cannot do anything with us
Open issues:
Diagram:


User selects "Register for Account"
User enters various contact information, billing information, and a username and password
Is all required information valid?
No
SHA-512 hash of the user's password is created, and all of their information is stored

Use Case: User Login
Primary actor: Users that wish to login to their Air-3550 account
Goal in context: To allow users to login to their Air-3550 account
Trigger: The user enters information into the login form on the homepage and clicks “Login”
Scenario:

1. User fills out form with appropriate information
    a. User enters their ID number
    b. User enters their password
2. User clicks “Login”
3. User login information is validated, and they are logged into their account, then shown their homepage

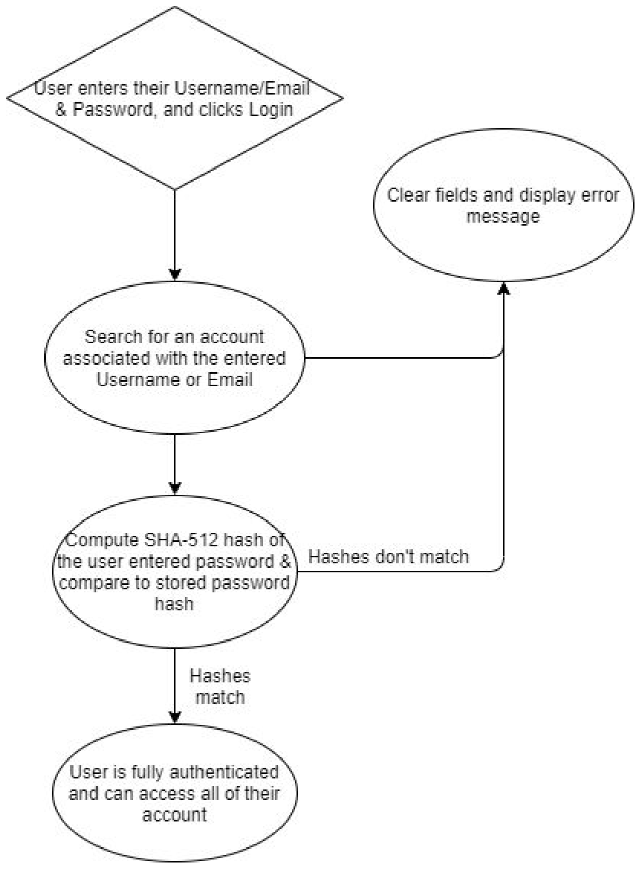
Exceptions:

1. If the user’s login information is invalid, they are shown an error message and the login form fields are cleared.

Priority: Highest priority, users must be able to login to the system to do anything else with us
Open issues:

1. Is security sufficient? Hacking into this feature would represent a major invasion of privacy

Diagram:


User enters their Username/Email & Password, and clicks Login
Clear fields and display error message
Search for an account associated with the entered Username or Email
Compute SHA-512 hash of the user entered password & compare to stored password hash
Hashes don't match
Hashes match
User is fully authenticated and can access all of their account

Use Case: Purchase Flight Ticket
Primary actor: Customer
Goal in context: To purchase a ticket or tickets for flights between two cities on given a day/days
Trigger: customer decides to purchase a plane ticket
Scenario:

1. Customer successfully logs in to their account
2. Customer selects a travel date (as well as return date for round trip)
3. Customer selects origin and destination city
4. System determines various routes with different times/connections/prices
5. Customer selects desired flight (or flights for round trip) and purchases ticket
    a. Ticket purchased with credit
    b. Ticket purchased with points - following Use Points use case
6. Purchase recorded in user's account as well as system
7. Customer receives confirmation of ticket purchase

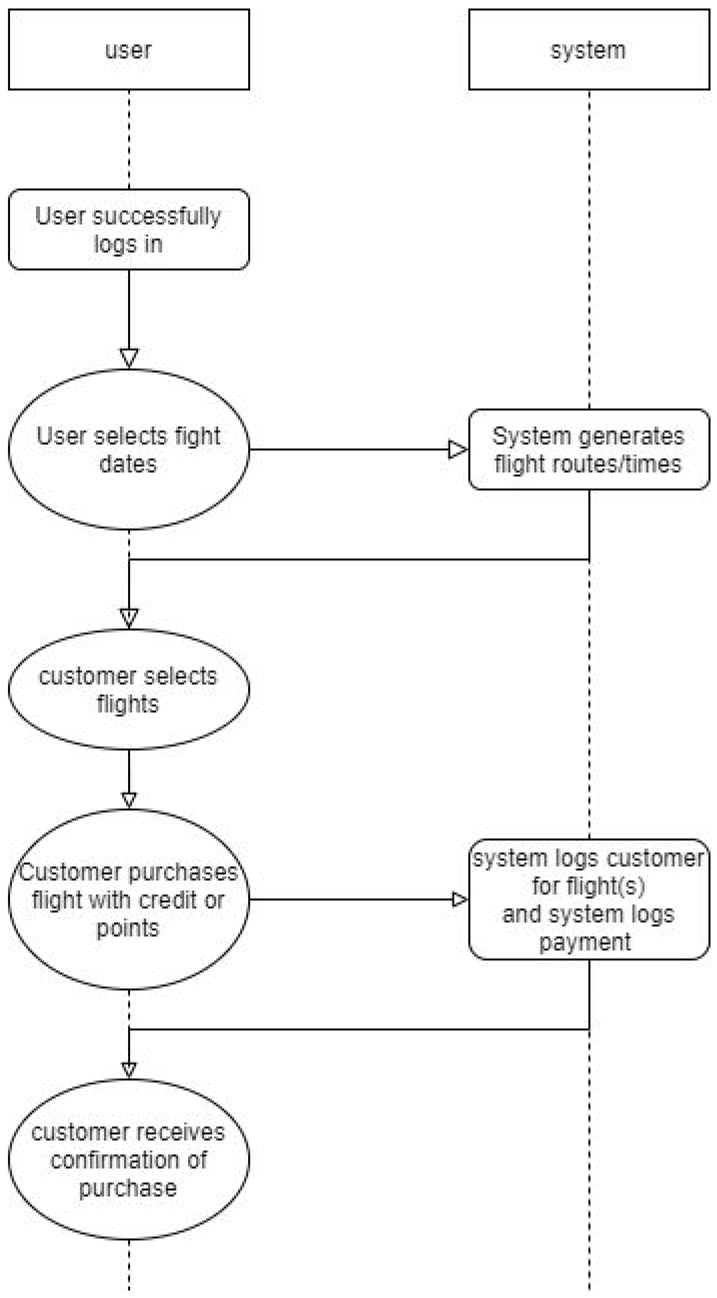
Exceptions:

1. Failed login
2. Insufficient credit or points
3. Customer attempts to book full flight
4. Customer attempts to book a flight more than 6 months in advance

Priority: low priority, implemented after a majority of functions are already in place
Open issues:

1. Performance, how well will system process flight requests and payment processing

Diagram:


user
system
User successfully logs in
User selects fight dates
System generates flight routes/times
customer selects flights
Customer purchases flight with credit or points
system logs customer for flight(s) and system logs payment
customer receives confirmation of purchase

Use Case: Determine plane for a flight
Primary actor: Marketing manager (MM)
Goal in context: determine which plane will be used for a flight so the system knows how many tickets to sell
Trigger: Marketing manager needs to select a plane for a given flight
Scenario:

1. MM successfully logs into account
2. MM selects option to choose plane for a flight
3. MM inputs flight date(s), origin city(ies), and destination city(ies)
4. System returns list of flight routes scheduled for the selected dates and cities
5. MM sets plane (from a list of available planes) for desired flight
6. MM repeats process as necessary

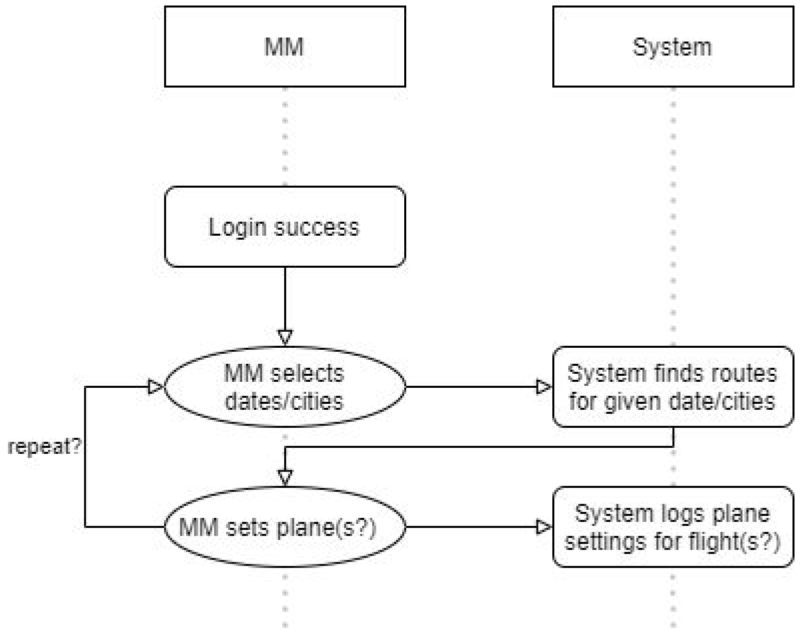
Exceptions:

1. Failed login
2. Desired flight not found

Priority: moderate-high, will need before customers can book, but will need to implement other basic functions first
Open issues:

1. Should a MM be able to change a plane that's already set? If so, how long before departure will they be allowed to do so? What if the plane is already fully booked?
2. How can this be implemented to be user friendly? Set multiple planes at once? Data to assist in choosing a plane?

Diagram:


MM
System
Login success
MM selects dates/cities
System finds routes for given date/cities
repeat?
MM sets plane(s?)
System logs plane settings for flight(s?)

Use Case: Determine routes
Primary actor: System
Goal in context: provide possible routes between two cities
Trigger: Any time a user needs to view possible routes
Scenario:

1. User requests routes between two cities on a given day
2. System gathers all flights for specific day
3. System generates routes fitting requirements:
    a. No route has more than three legs (2 connections)
    b. No connection time is greater than 40 minutes
4. Display generated flight routes flight routes

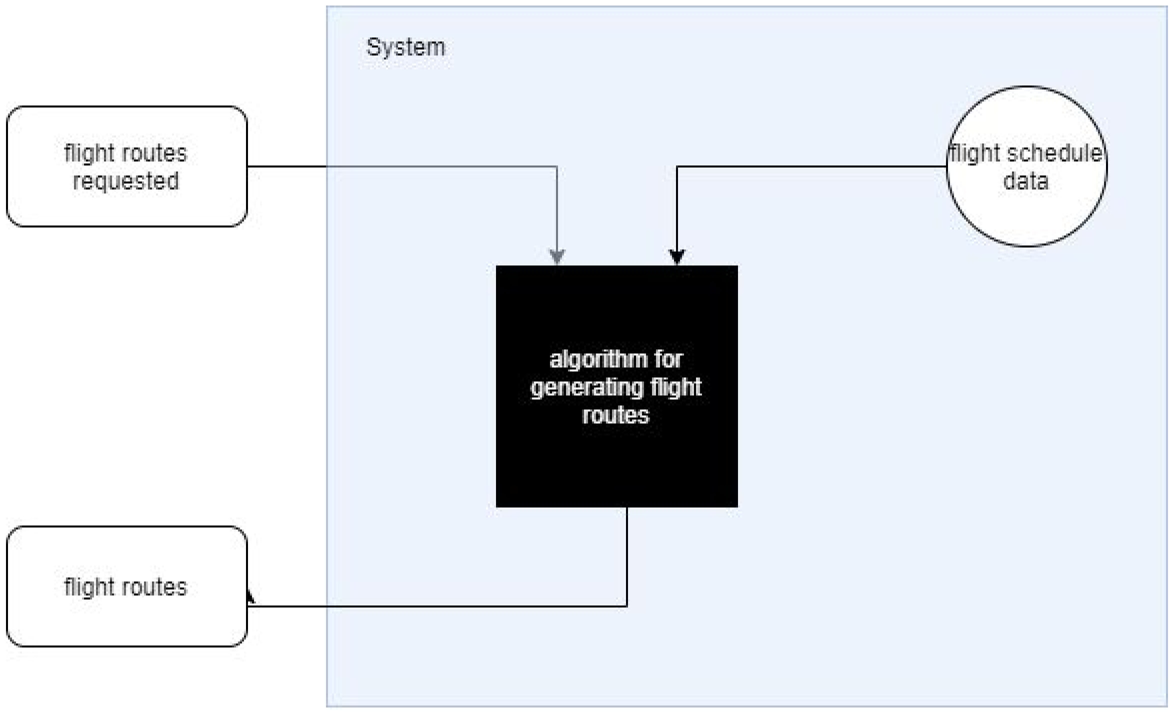
Exceptions:

1. No route generated
2. One or both cities not in system
3. No flights scheduled for that day

Priority: High, needed in multiple other user cases. Need to be able to determine routes before customers can purchase tickets
Open issues:

1. Performance, how quickly can we generate flight routes?

Diagram:



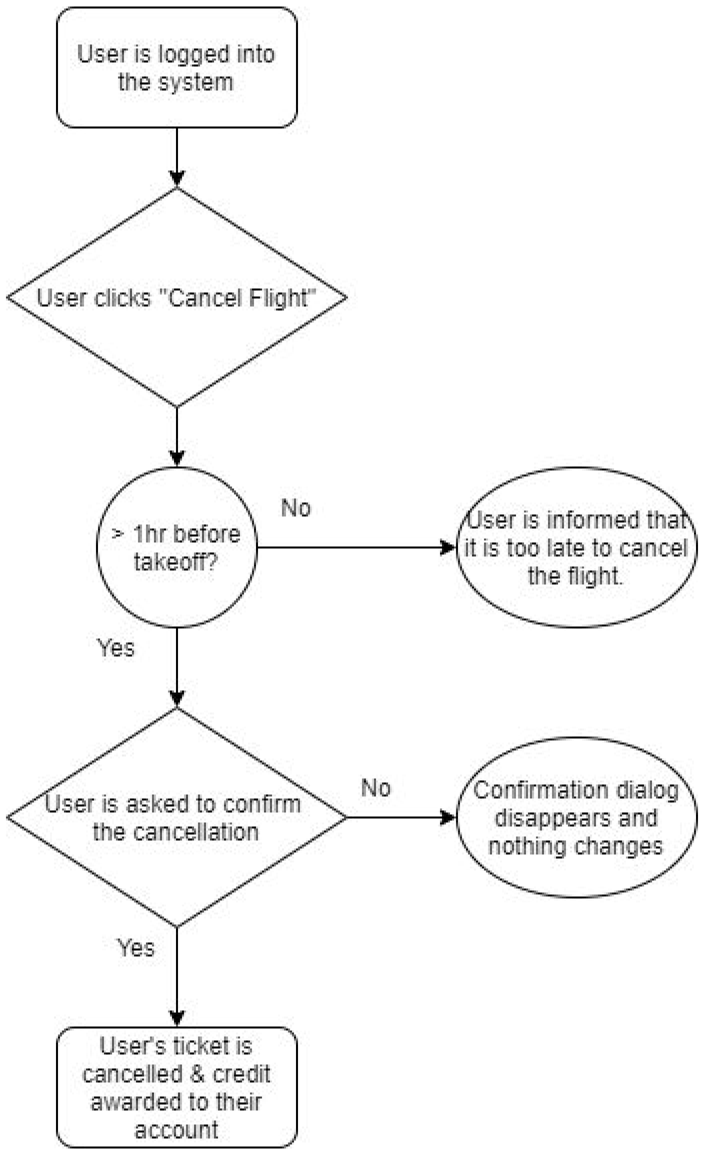
Use Case: Cancel Flight Ticket
Primary actor: Customer who has a ticket purchased for a flight
Goal in context: To allow customers to cancel tickets for flights that are at least 1 hour from departure
Trigger: Customer presses the "Cancel Flight" button, on the appropriate flight on their account page
Scenario:

1. Customer is successfully logged into their account & on their account page
2. Customer clicks "Cancel Flight" on the flight they wish to cancel
3. Customer is asked to confirm that they would like to cancel their ticket, and shown all information about the flight
4. Their ticket is cancelled for the flight
5. The user is awarded appropriate credit to their account - see use case "Award Credit"

Exceptions:

1. The flight is scheduled to take off in less than 1 hour
2. Customer doesn't confirm the cancellation

Priority: Low priority, most features can be implemented first, users must be able to book flights before they can cancel them.
Open issues:
Diagram:


User is logged into the system
User clicks "Cancel Flight"
> 1hr before takeoff?
No
User is informed that it is too late to cancel the flight.
Yes
User is asked to confirm the cancellation
No
Confirmation dialog disappears and nothing changes
Yes
User's ticket is cancelled & credit awarded to their account

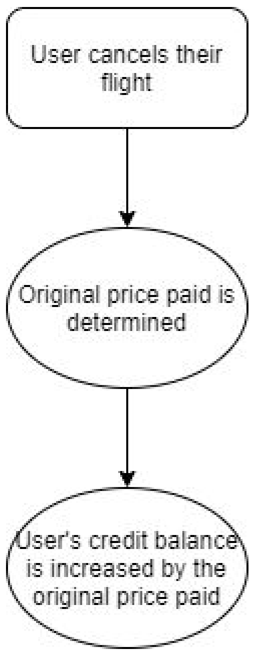
Use Case: Award Credit
Primary actor: System
Goal in context: To award credit to users who have cancelled their flight ticket
Trigger: A user's flight ticket is cancelled
Scenario:

1. User's flight ticket is cancelled
2. Amount of credit is determined by price paid for original ticket
3. User's credit amount is increased by the price paid for original ticket

Exceptions:
Priority: Low priority, users must be able to cancel flights first
Open issues:

1. How should we handle awarding credit if users purchased their ticket with points? Will we need to award points instead?

Diagram:


User cancels their flight
Original price paid is determined
User's credit balance is increased by the original price paid

Use Case: Award Points
Primary actor: System
Goal in context: To award points to the user when a flight they paid for takes off, whether or not they are on it
Trigger: A flight the user has a paid ticket for takes off, whether or not the user is on it
Scenario:

1. A flight takes off that the user has a paid ticket for
2. Original price paid for the user's ticket is determined
3. User's points balance is increased by .01 * the original price paid for the ticket

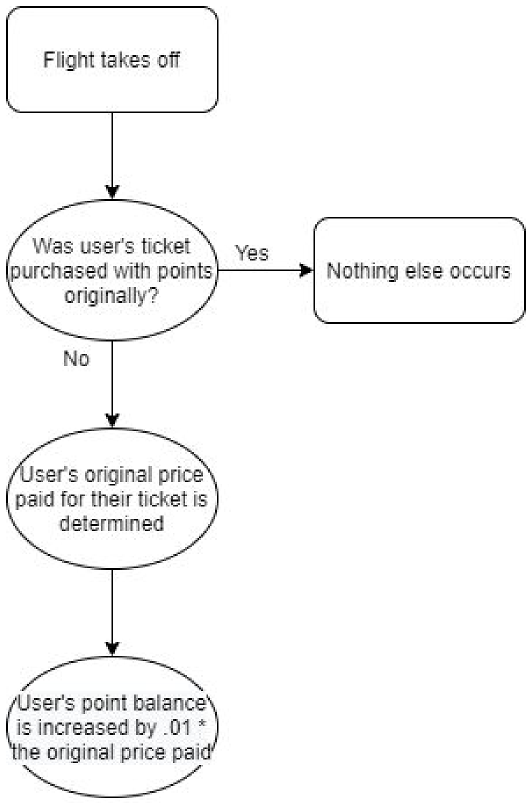
Exceptions:

1. User purchased the flight ticket with points originally
2. User cancelled the ticket with > 1hr remaining before takeoff

Priority: Low priority, users must be able to purchase flight tickets before they can be awarded points
Open issues:

1.

Diagram:


Flight takes off
Was user's ticket purchased with points originally?
Yes
Nothing else occurs
No
User's original price paid for their ticket is determined
User's point balance is increased by .01 * the original price paid

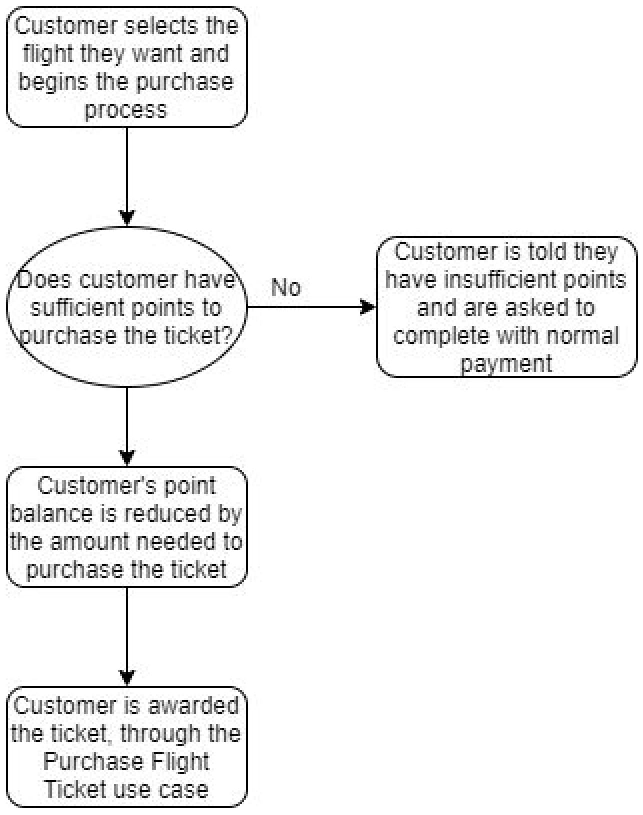
Use Case: Use Points
Primary actor: Customer
Goal in context: Customer wishes to purchase a flight ticket using points
Trigger: Customer is purchasing a flight ticket and has enough points to do so
Scenario:

1. Customer selects the flight they would like to take & begins the purchase process as detailed in the Purchase Flight Ticket use case.
2. If customer has 100x more points than the cost of their flight, i.e. 10,000 points for a $100 ticket they may use their points to purchase the ticket
3. Customer's point balance is reduced by the amount needed to purchase the ticket
4. Customer is awarded the ticket through the Purchase Flight Ticket use case

Exceptions:

1. Customer doesn't have enough points to purchase the ticket

Priority: Low priority, customers must be able to earn points before they can purchase tickets with him
Open issues:
Diagram:


Customer selects the flight they want and begins the purchase process
Does customer have sufficient points to purchase the ticket?
No
Customer is told they have insufficient points and are asked to complete with normal payment
Customer's point balance is reduced by the amount needed to purchase the ticket
Customer is awarded the ticket, through the Purchase Flight Ticket use case

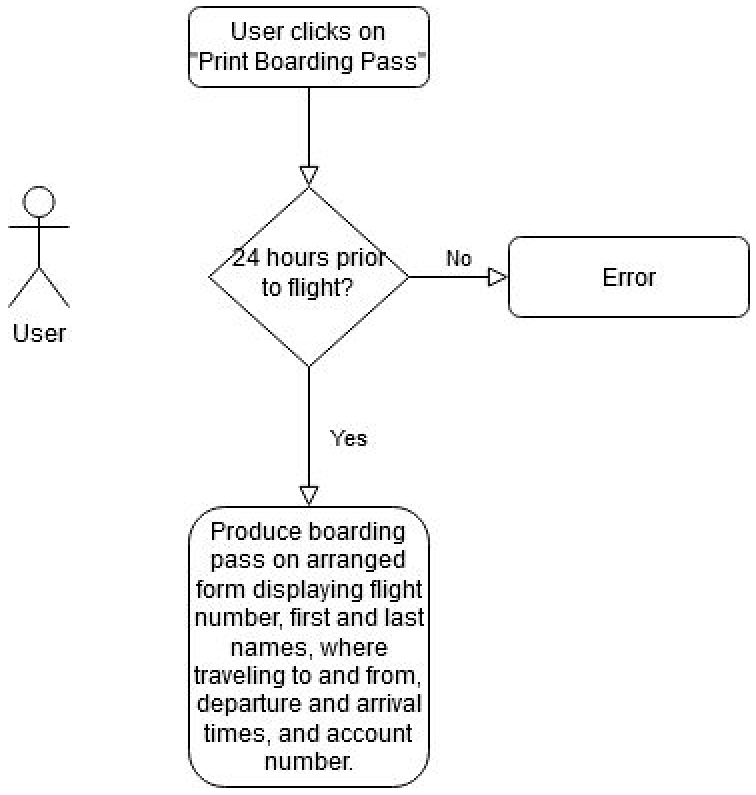
Use Case: Print Boarding Pass
Primary actor: Customer
Goal in context: Customer wishes to print their boarding pass
Trigger: Customer has paid for a flight and is ready to print their boarding pass
Scenario:

1. Customer clicks on “Print Boarding Pass”.
2. A printable window pops up that has their flight information that includes: their flight number, first and last names, where they are traveling from and to, their departure and arrival times, and their account number.

Exceptions:

1. Customer has not purchased a flight.
2. It is not 24 hours prior to their flight time.

Priority: High priority. The customer must be able to print their boarding pass in order to take their flight that they paid for.
Open issues:
Diagram:


User clicks on "Print Boarding Pass"
User
24 hours prior to flight?
No
Error
Yes
Produce boarding pass on arranged form displaying flight number, first and last names, where traveling to and from, departure and arrival times, and account number.

Use Case: Record Financial Transaction
Primary actor: System
Goal in context: To keep track of financial transactions used in the purchasing of flight tickets
Trigger: A customer decides to purchase a flight ticket
Scenario:

1. Customer purchases flight ticket
2. System checks if the flight ticket was purchased with money only, not credit or points
3. System records customer's name, credit card number, and the amount to charge

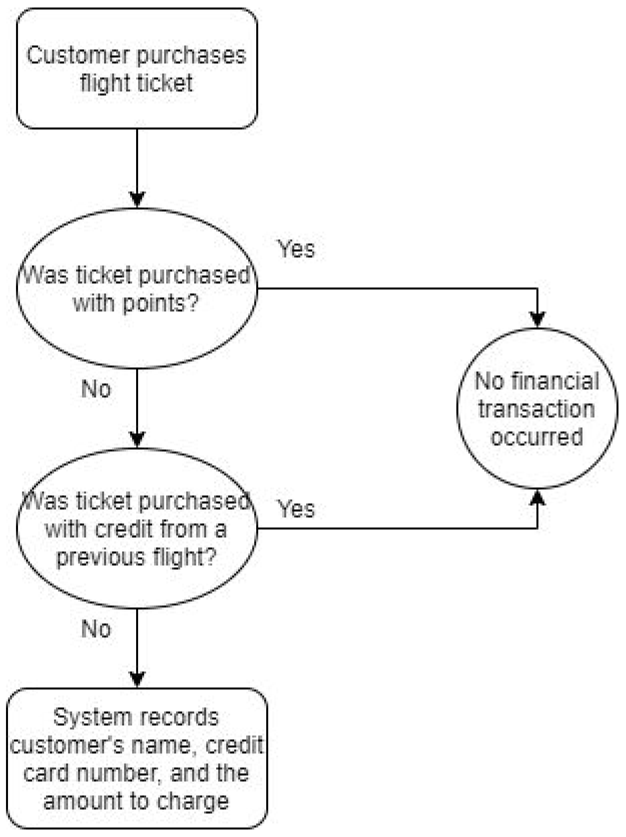
Exceptions:

1. The user purchased the flight ticket with points
2. The user purchased the flight ticket solely with credit from a previous cancelled ticket

Priority: Medium Priority, needs to be implemented before users can purchase flight tickets
Open issues:

1. We need to ensure that the security of these financial records is strong enough. Any breach of this information would be incredibly damaging to customer privacy.

Diagram:


Customer purchases flight ticket
Was ticket purchased with points?
Yes
No
No financial transaction occurred
Was ticket purchased with credit from a previous flight?
Yes
No
System records customer's name, credit card number, and the amount to charge

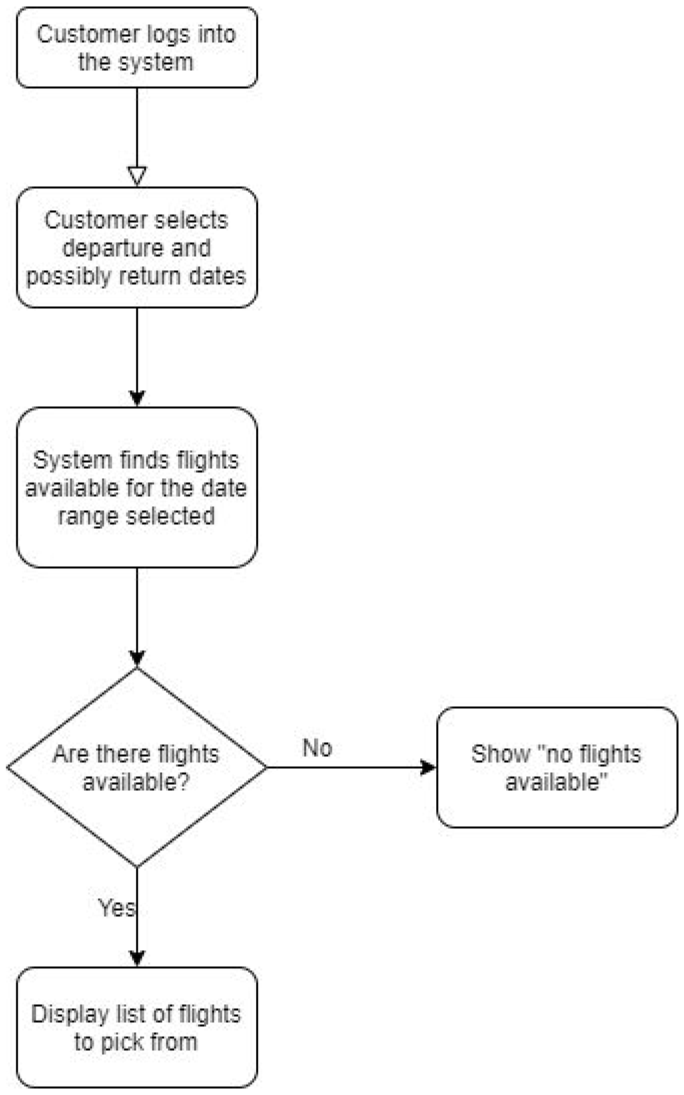
Use Case: Select Travel Date(s)
Primary actor: Customer
Goal in context: Customer is looking for a flight and has entered their departure and return dates to which the system responds by listing available flights
Trigger: A customer needs to find flights for a date (one-way) or dates (round-trip)
Scenario:

1. Customer logs in to our system.
2. Customer enters departure date and possibly return date for round-trip.
3. The system displays flights available for the entered date(s).
4. The customer is able to book their flight.

Exceptions:

1. There are no flights for the selected date(s).
2. The planes are all booked.

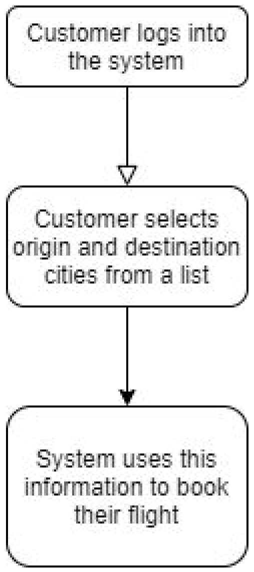
Priority: Very high. This is essentially the first step for our customers using our airline.
Open issues:
Diagram:


Customer logs into the system
Customer selects departure and possibly return dates
System finds flights available for the date range selected
Are there flights available?
No
Show "no flights available"
Yes
Display list of flights to pick from

Use Case: Select Origin & Destination Cities
Primary actor: Customer
Goal in context: Customers must be able to select from a list of origin and destination cities in order to book their flight
Trigger: Customer wants to book a flight and needs to select their origin and destination cities
Scenario:

1. Customer logs into their account.
2. Customer selects from a dropdown list their origin and destination cities.
3. From this information, along with the info from "Select Travel Dates," the customer can book their flight.

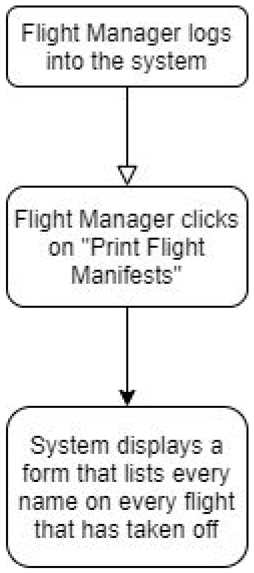
Exceptions:
Priority: High. They must be able to select origin and destination cities in order to use our service.
Open issues:
Diagram:


Customer logs into the system
Customer selects origin and destination cities from a list
System uses this information to book their flight

Use Case: Print Flight Manifests
Primary actor: Flight Manager (FM)
Goal in context: Flight manager needs to be able to print flight manifests that lists everyone on each flight when it takes off
Trigger: FM logs into the system and clicks on “Print Flight Manifests”
Scenario:

1. FM logs into the system.
2. FM needs to view the current flight manifests.
3. FM clicks on “Print Flight Manifests”.
4. The system brings up a printable form that displays a list of each name on each flight when it takes off.

Exceptions:
Priority: Extremely high. This is a necessary part of the airline market for several reasons.
Open issues:
Diagram:


Flight Manager logs into the system
Flight Manager clicks on "Print Flight Manifests"
System displays a form that lists every name on every flight that has taken off

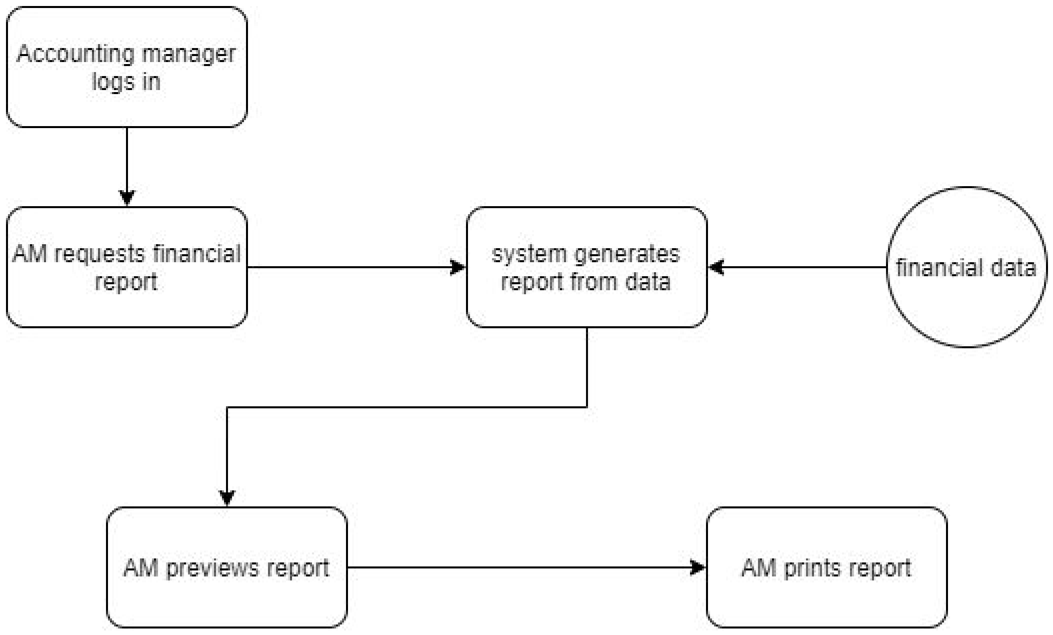
Use Case: Print Accounting Report
Primary actor: Accounting Manager (AM)
Goal in context: Provide AM with a summary of accounting info
Trigger: AM decides to print accounting report
Scenario:

1. AM logs into account
2. AM requests Accounting report
3. System gathers accounting records
4. System generates report from record
5. AM views preview of report
6. AM prints report

Exceptions:
Priority: low, customers need to be able to purchase tickets before the AM can view how much money the company makes.
Open issues:

1. Performance. How long will it take to generate a report? Sales data changes frequently, how will the system keep up?

Diagram:


Accounting manager logs in
AM requests financial report
system generates report from data
financial data
AM previews report
AM prints report

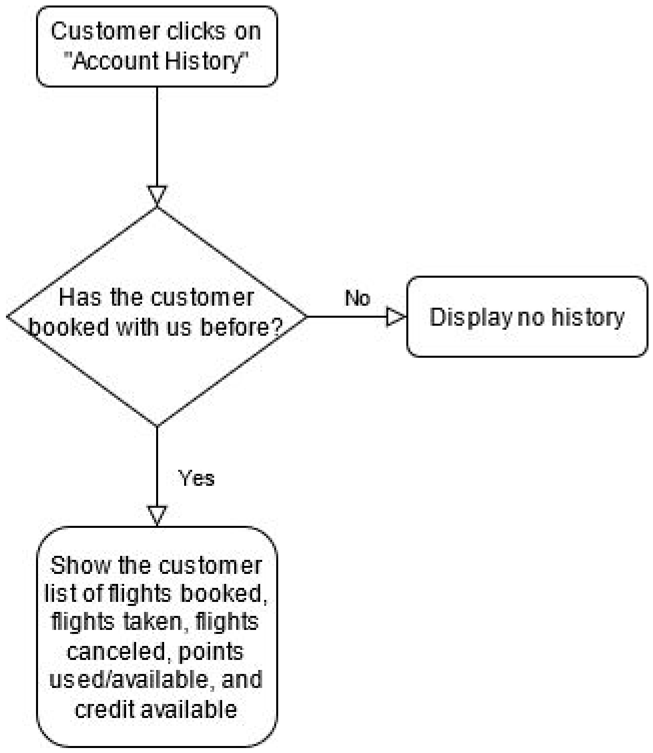
Use Case: View Account History
Primary actor: Customer
Goal in context: Provide the customer with a record of their account history
Trigger: Customer wants to view their account history
Scenario:

1. Customer logs into account.
2. Customer clicks on "Account History".
3. A form appears that has information such as: flights booked, flights taken, flights canceled, points used, and points available (not mentioned in Problem Statement, but maybe include credits from flights canceled).

Exceptions:

1. Customer has no history with us.

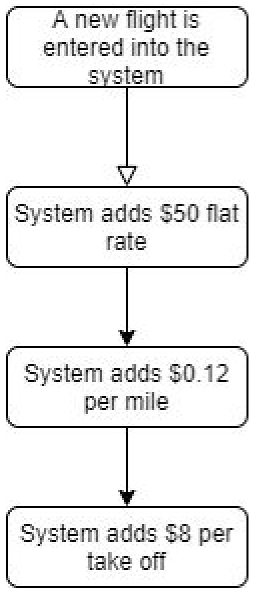
Priority: Low priority. It is a convenience to the customer, but it doesn't make or break our business.
Open issues:
Diagram:


Customer clicks on "Account History"
Has the customer booked with us before?
No
Display no history
Yes
Show the customer list of flights booked, flights taken, flights canceled, points used/available, and credit available

Use Case: Determine Flight Pricing
Primary actor: System
Goal in context: System needs to calculate the price of tickets accurately for customers and our business
Trigger: Flights are entered into the system by a load engineer and the price needs to be calculated based on this information for each flight.
Scenario:

1. A new flight is entered into the system.
2. System calculates price based on several factors.
3. System includes a $50 flat rate for all flights.
4. System adds $0.12 per mile.
5. System adds $8 per take off for federal fees to pay for TSA agents.

Exceptions:
Priority: Very high. We need this information before we can book flights.
Open issues:
Diagram:


A new flight is entered into the system
System adds $50 flat rate
System adds $0.12 per mile
System adds $8 per take off

Use Case: Customer updates account information
Primary actor: Customer
Goal in context: update account information and/or password for customer's account
Trigger: customer decides to update information and/or password
Scenario:

1. Customer logs into account
2. Customer selects option to change account info
3. Customer selects information to be changed
    a. If customer selects to change password, system asks for old password and verifies it before proceeding
4. Customer enters new information
5. System updates user info and creates new hash for password if password was changed

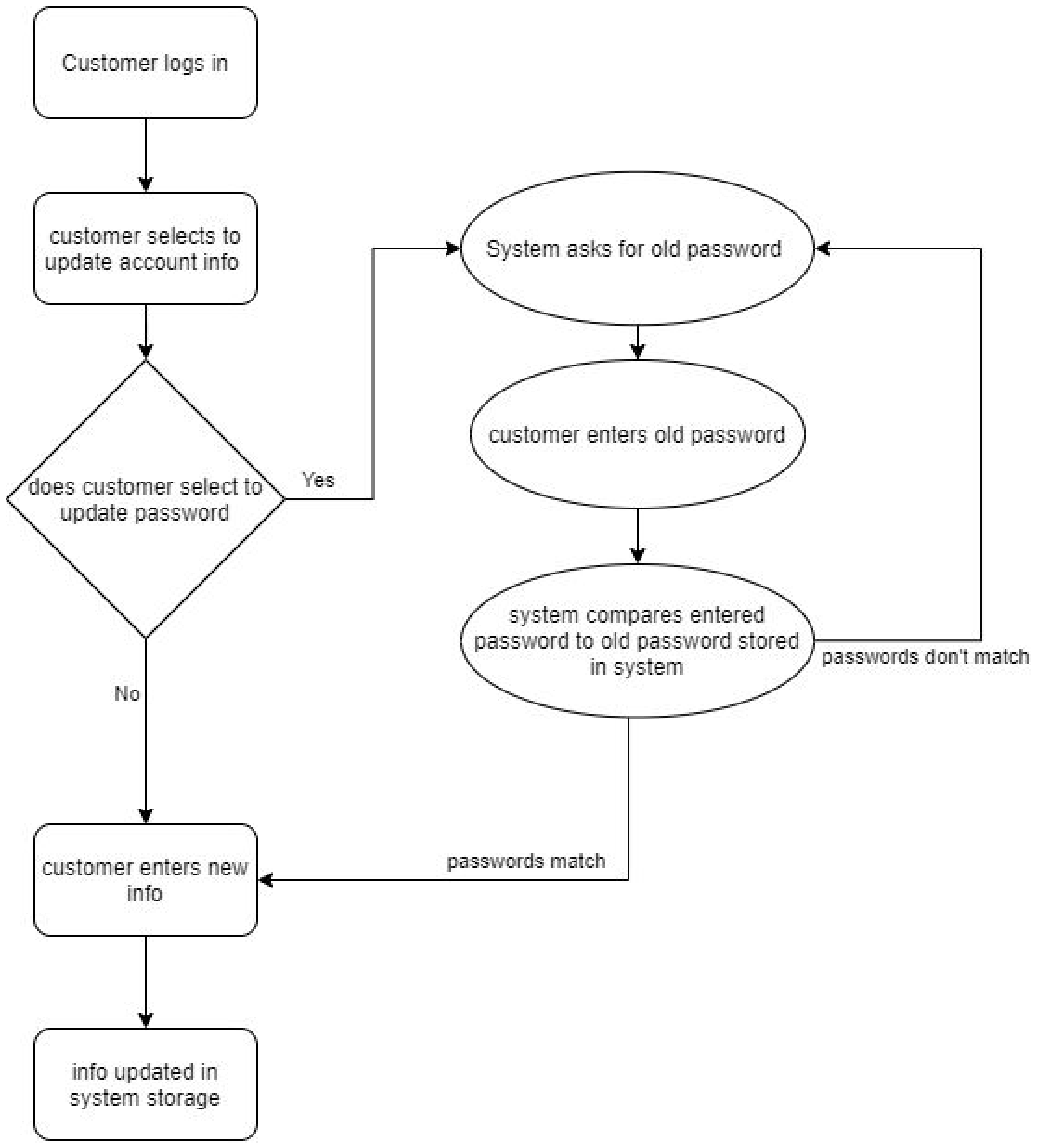
Exceptions:

1. Customer's new password is same as old password

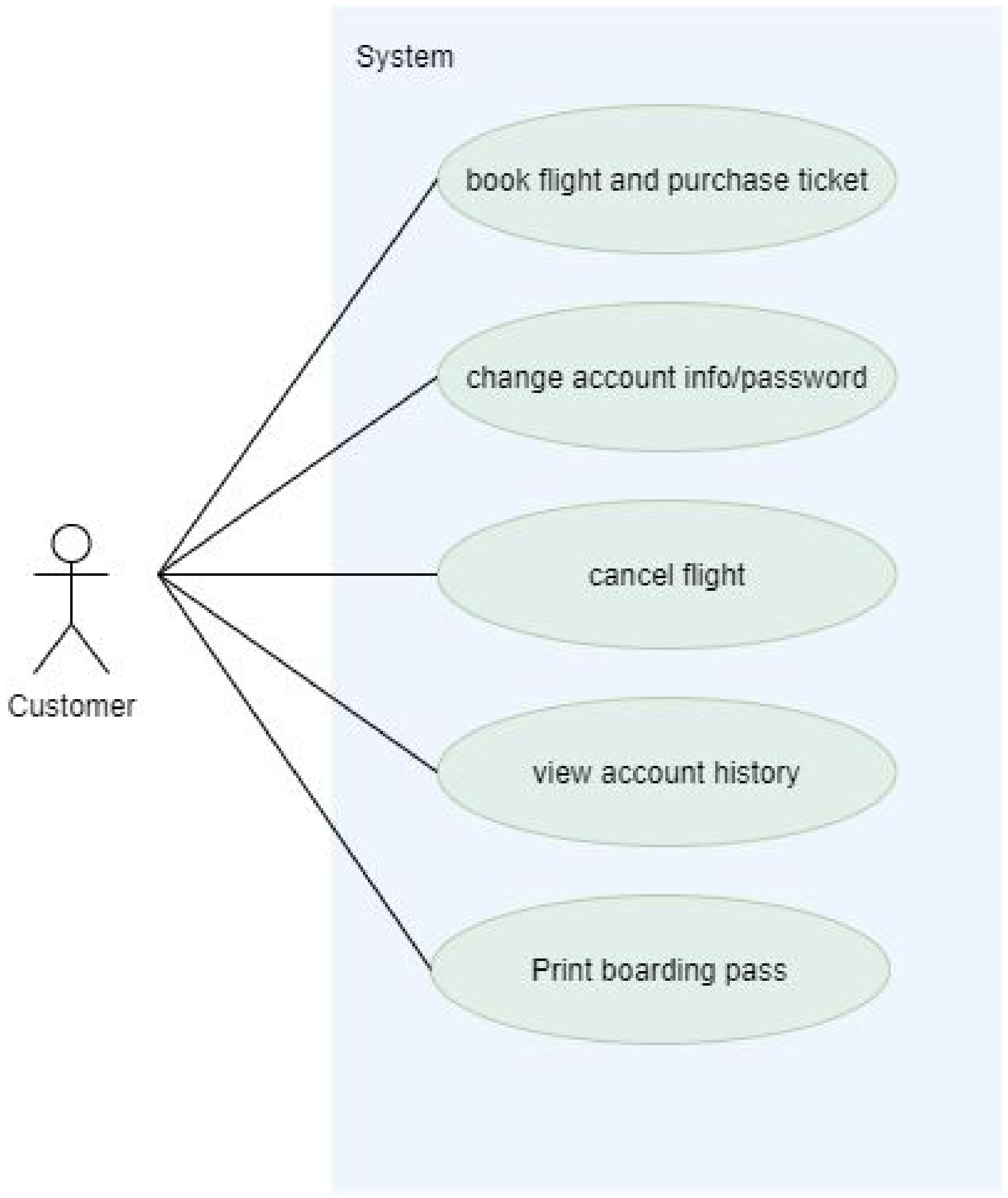
Priority: low-moderate, basic functions must be implemented first. Important for customer but not vital
Open issues:

1. How often should customers be allowed to change info? What are other security implications of changing password?

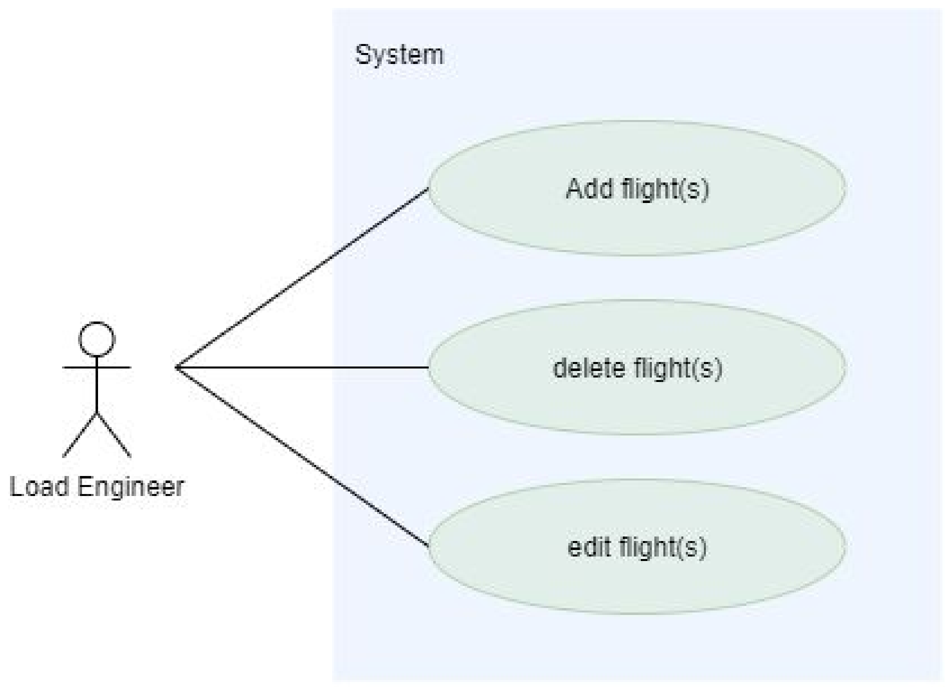
Diagram:


Customer logs in
customer selects to update account info
does customer select to update password
Yes
No
System asks for old password
customer enters old password
system compares entered password to old password stored in system
passwords don't match
passwords match
customer enters new info
info updated in system storage

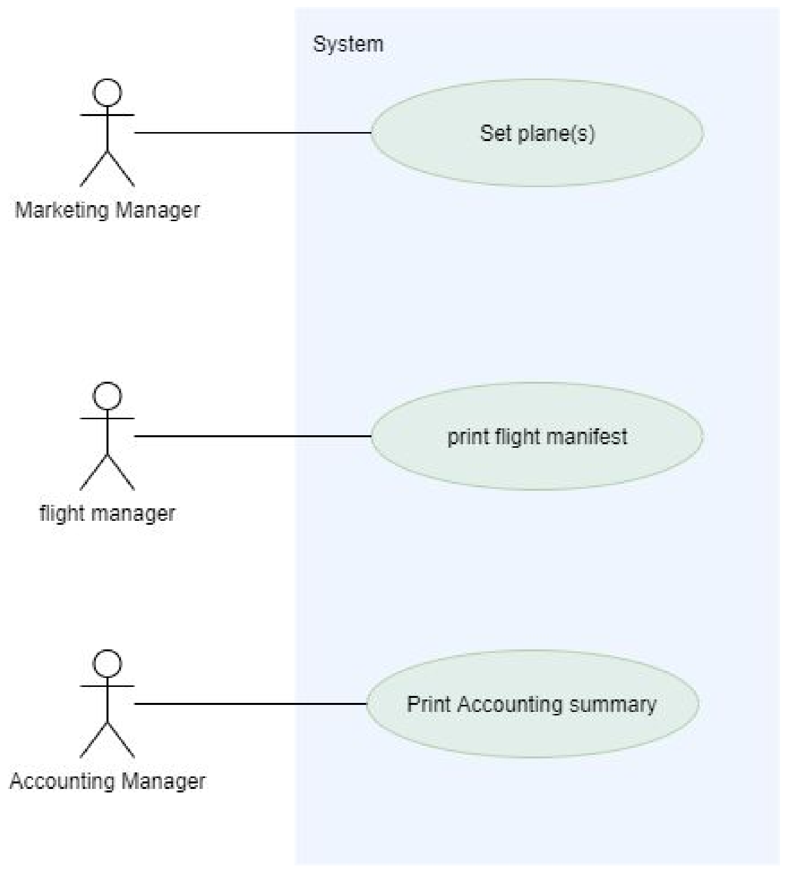
Diagrams for functions accessible to each type of user
Customer:



Load Engineer:



Managers:



Other requirements not covered in a use case:

1. List of real-world cities with real airports served by our airlines
    a. “Collapse” New York, Chicago, etc. locations into a single airport
    b. Must service at least 10 airports
        i. Must contain Nashville & Cleveland
    c. We must know the straight-line distances between airports
2. Won’t have direct flights from everywhere to everywhere
    a. Some flights, i.e. Nashville => Seattle, will need a connection, perhaps in Chicago
    b. Flights may not have more than 3 legs, or 2 connections
    c. Connections / layovers must be at least 40 minutes long
3. We need to have a variety of planes
    a. At least three from the 737, 747, 757, 767, 777 models
        i. From these, each flight will have the appropriate capacity for their chosen plane
        ii. Some flights, i.e. LA => New York will prefer a bigger plane like a 747, while shorter flights will need the 737