Flight Booking System

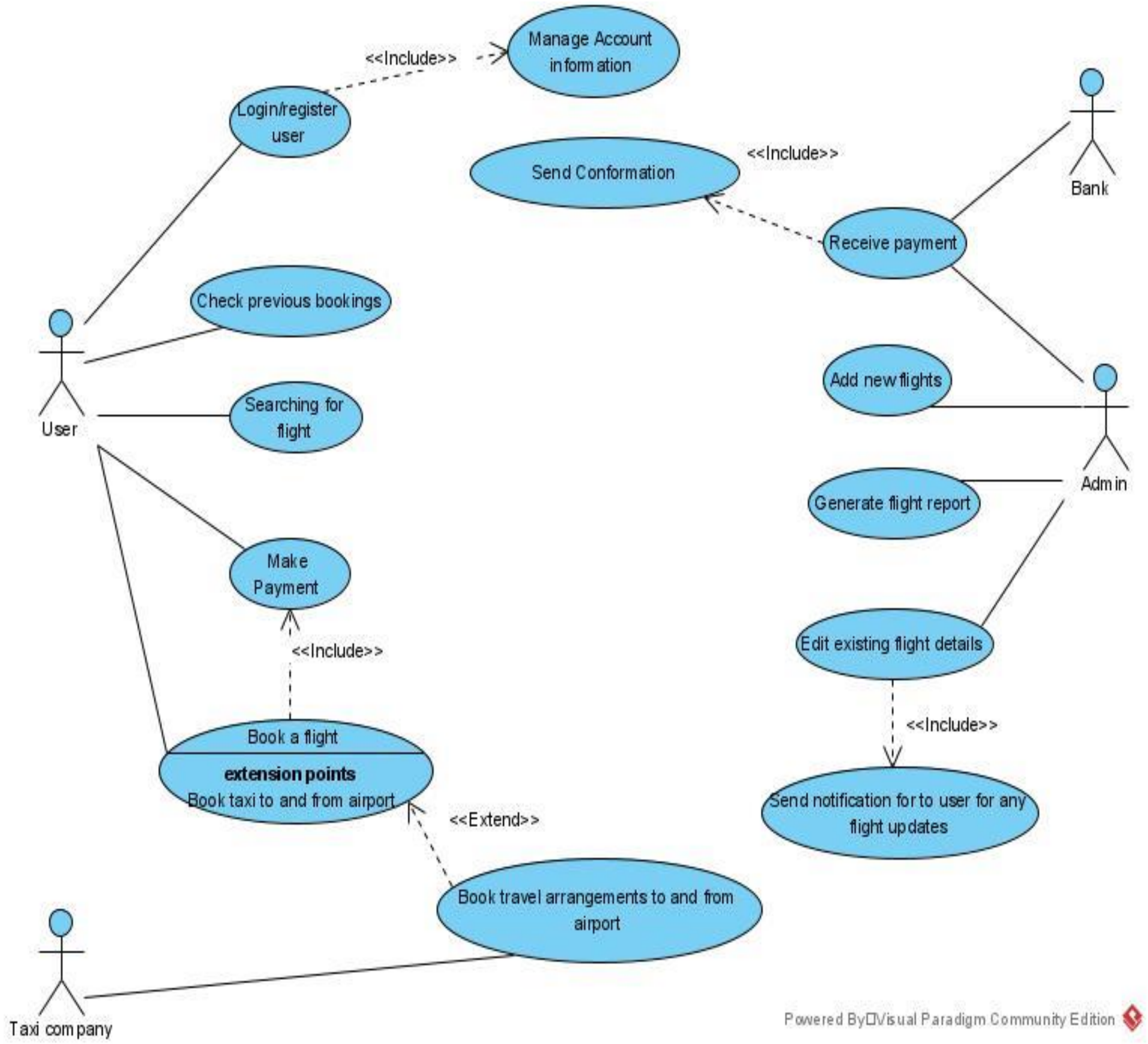
Group 28-

- Inderpreet Bahi
- Sam Thandi
- Rohan Bhuhi
- Imran Lawal
- Leon Skuqi

Functional Requirements:

1. A user must be able to book a flight using their name and email address.
2. Users must be able to create an account on the website.
3. The user must be able to pay.
4. Users can see and manage previous bookings.
5. Users will be able to search for flights.
6. The system must show the user what flights are available.
7. Users must be able to opt in or out of notifications on specific flights and subscribe to them.
8. The system will send out email/mobile push notifications to users regarding updates on their subscribed to flights.
9. The system administrator can add new flights.
10. The system administrator can edit existing flight details.
11. The system administrator must be able to generate flight reports for all flights.

Use Case Diagram:



Use Case Descriptions:

Use Case: Make Payment
Author: Rohan Bhuhi
Date: 03-03-2021

Purpose: To make a payment for a flight booking and addons

Actors: Customer and Bank

Overview: The use case starts when the customer selects checkout option after selecting all their flight details and add-ons. The system requests the user preferred payment method. The system then requests the information required for the specific payment method. The system checks if the information is valid, and funds are available (checking with banks or PayPal etc.). If the validation is successful, the customer can confirm payment. If validation fails, the system offers to try again or choose a different payment method. Once payment is confirmed the system must send a payment confirmation and receipt by email to the customer.
Pre-Condition:
– The flight details of flight, time, date, and destination must be complete and added to basket.
Post-Condition:
– The flight must be booked and reserved for the person booking along with any add-ons.
– The payment confirmation and receipt must be sent to the customer.
– The money must be paid by the customer to the relevant companies (flight, taxi or car hire)

Actor actions	System actions
1. Begins when a customer confirms checkout	
2. Customer selects payment method	3. The system requests user details
4. Customer enters details and confirms	5. The system check payment details and funds with banks or PayPal etc.
	6. The system asks user to confirm the payment
7. Customer confirms payment	8. The system confirms the customer flight and any add-ons with the companies
	9.The system send a confirmation email and receipt to the Customer

Alternative flow of events:
– Step 3: Customer authorization failed. Display an error message, cancel the transaction, and eject the card.
– Step 7: Customer has insufficient funds in its account. Display an error message and go to step 6.
Exceptional flow of events:
– Website failure in the process of the transaction before step 9, cancel the transaction

Use Case: Book a flight
Author: Rohan Bhuhi
Date: 03-03-2021

Purpose: To book a flight
Actors: Customer

Overview: The use case starts when the customer selects the flight. The system then offers the customer additional services to add to the flight booking (Extra luggage, taxi or car hire etc.). The Customer will select any if required. The case ends when the customer presses checkout.

Pre-Condition:
– The flight details of flight, time, date, and destination must be complete and added to basket.
Post-Condition:
– The flight must be added to the basket along with any add-ons.
– The flight must be reserved for the customer until payment is made booking along with any add-ons.

Actor actions	System actions
1. Begins when a customer selects a flight	
2. Add flight(s) to basket	3. Reserve flight
	4. Offers addons
4. Selects addons (luggage, taxi or car hire etc.)	
5. Add addon(s) to basket	6. Reserve addon(s)
7. Checkout	

Alternative flow of events:
– Step 4: Customer has no addons. Go straight to from step 3 to 7.
– Step 3 or 6: Customer takes longer than 1 hour since adding to
basket. Display a timeout message and go to step 1.
Exceptional flow of events:
– Website failure in the process of the transaction before step 7, cancel the transaction.

Use Case: Users finding and confirming a flight.

Author: Sam Thandi
Date: 3/3/2021

Purpose: To make a payment for a flight booking and addons

Actors: Registered and unregistered customers.

Overview: The use case starts when the user selects either one of our search engines for flights (either our search bar or our geographical map feature). The system takes the user input of where the user wants to fly to and fly from and matches their request with specific flight details that relates to the desired parameters. Once the desired flight is selected, the system shows the user details about this specific flight and compares them to other similar flights offered by similar companies. The system will then check if flights are fully booked or not, if they are fully booked, the system will allow the user to opt into notifications to inform the user if the flight becomes available. If the user accepts to opt into notifications, the system must email them whenever there is an update about availability with the given flight. If the flight is not full, the user can then select the given flight and move onto booking.

Pre-Condition:

- The system must be able to search for any valid flight or any valid flight path specified by a given user.
- The system will allow users to search for completely booked flights and allow users to opt into notifications about those singular flights.

Post-Condition:

- The flight has been selected.
- The system saves the users search history.
- The system sends an email to the user letting them know they opted into notifications.
- The system will send an email if the previously fully booked flight is now available.

Actor actions	System actions
	1.The system asks the user for departure date/airport to depart from, desired destination, number of people going on the flight, etc.
2.Begins when a user decides to search for a flight with the parameters of departure date, arrival destination, number of people, etc.	3.The system checks all flights with the desired attributes in the flight database and outputs the list to the user
4.User can decide if they want to opt into notifications for fully booked flights once they become available	5.The system adds the user to notifications for flight x database.
6.The user selects which flight they want to take out of the suggested flights.	7.The system takes the user to the desired flight page and saves the users search history
8.User chooses to confirm flight and move onto booking phase.	

Alternative flow of events:

- Step 3: There are no available flights that correspond to the users desired flight, as a result the system will output “No flights found under your current parameters” to the user.
- Step 5: If the user is unregistered the system will prompt the user to either create an account or enter in their email without creating an account yet.

Exceptional flow of events

- System failure on the website before step 8 causing a cancellation in their flight booking.

Use Case: Edit existing flight details

Author: Inderpreet Bahi

Date: 10/3/2021

Purpose: To allow an admin to adjust flight details for an upcoming flight.

Actors: Admin

Overview: The use case starts when the admin searches for an upcoming flight. The admin then selects the flight that needs to be changed and can view all their details such as the airline, arrival destination, time of flight, time of departure and landing, the manifesto of what seats have been taken and their corresponding ticket holders and departure destination. Once the desired flight is selected, the admin can update the details of the flight. Once the admin is done making changes, they must confirm that the flight details have been updated accordingly. Once the system receives confirmation users that are currently holding a ticket and have opted to receive notifications for that flight will be sent notification of all changes that have been made and information that may affect the.

Pre-Condition:

- The system must be able to search for any valid flight specified by the admin.
- The system must have generated a flight report/manifesto so the admin can see what users hold which tickets currently.
- The system must have a generated data base of all the previous changes made to a particular flight.

Post-Condition:

- The flight/flights have been selected.
- The system saves the history of all changes made to the flight.
- The system must send an email to any users that are associated with the changes.

Actor actions	System actions
1. Begins when admin searches for what flights they want to make changes to via flight number.	2. The system searches the data base for the upcoming flights
4. Admin selects what changes they want to make.	3.The system then generates the flights details along with the current manifesto for the flight
6.Admin confirms the changes to be made.	5. System ask the Admin for confirmation of changes.
	7.System receives confirmation and stores the changes and updates the data base
	8. System creates/ updates the history of all changes that have been made to the selected flight
	9. System then sends notification to any users that currently hold a ticket for the flight.

Alternative flow of events:

- Step 1: There are no flights that correspond to the entered flight ID number, as a result the system will output “No flights found with that ID number, please try again” to the Admin.
- Step 5: The system can identify if the Admin makes an impossible change to the flight details such as changing the time of departure to 23:77. The system will then notify the admin of the error until it is resolved before moving onto step 6

Exceptional flow of events

- If system fails whilst making changes, all changes will be reverted according to the backlog of history of the changes.

Use Case: Login

Author: Imran Lawal

Date: 3/3/21

Actors: User

Purpose: To allow users to login safely with their email and password

Overview: The use case starts when A registered user tries to sign back in the system. The system requests the users email and password. The system validates the user's login details. If the validation is succeeded, then the user will then be able to access information such as being able to check current and past booking systems. But before this the user is taken to the home page.

Pre-condition:

- User must have an already existing account.
- They must be on the application to be able to login.
- User must not be already logged in.

Post – condition:

- Both login details should match database information on user's login details

Actor actions	System actions
1. Begins when a customer is on the application and would like to log in to their account	
2. Customer selects the login button	
4. Customer enters their login details	5. System verifies user's personal details
	6. System fetches user information
	7. System then displays user information

Alternative flow of event:

- Step 5: Customer authorization failed. Displays and error message and asks user to input their details again.
- Step 6: System fails to gather user information. Error message will be displayed and will ask user to enter user details again.

Exceptional flow of events

- System failure on the website before step 7 causing a cancellation in their login

Use Case: Register

Author: Imran Lawal

Date: 3/3/21

Actors: User

Purpose: To allow the user to create an account with us allowing the user to access new features and store his/her information about previous or present booked flights.

Overview: This use case is initiated when A unregistered user would like to create an account with us. When the user arrives at our page, he/her will be able to register by selecting the register button, which will take the user to where he/her can create an account by entering A valid password which satisfies all requirements asked by the system along with entering an existing valid email.

Pre – condition:

- User must not already have an account.
- Must have a valid email to register with

Post – Conditions:

- User details being entered to create account must match requirements such as password length checks, special character use, and such.

Actor actions	System actions
1. Begins when A user has not previously created an account before and selects register button.	2. System asks user to create password
3. User inputs email and password to be created	4. System checks if password is valid
	5. System appends user details to current database

Alternative flow of events:

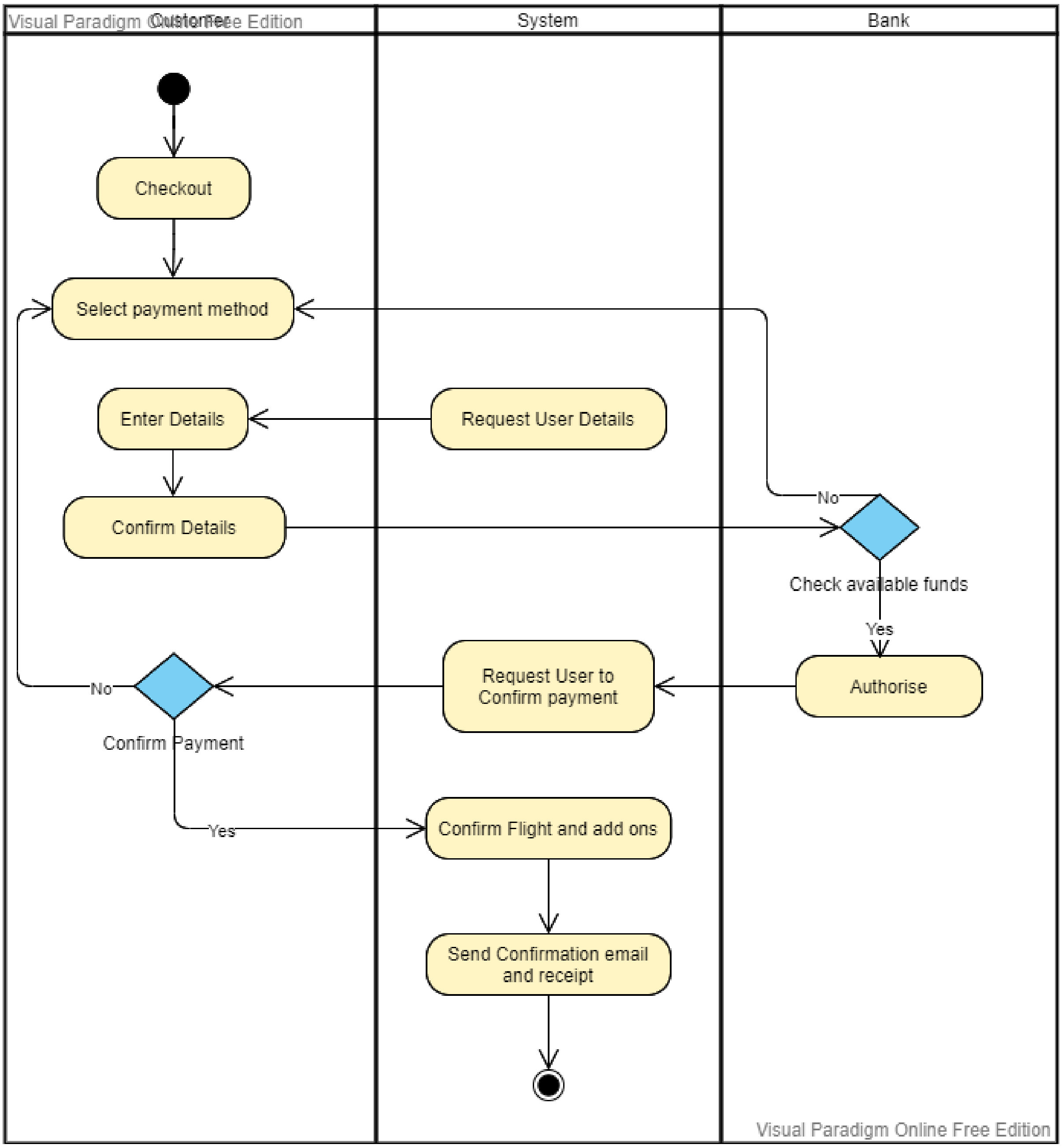
Step 4: Password is invalid. System displays error message and asks user to come up with A new password.

Exceptional flow of events

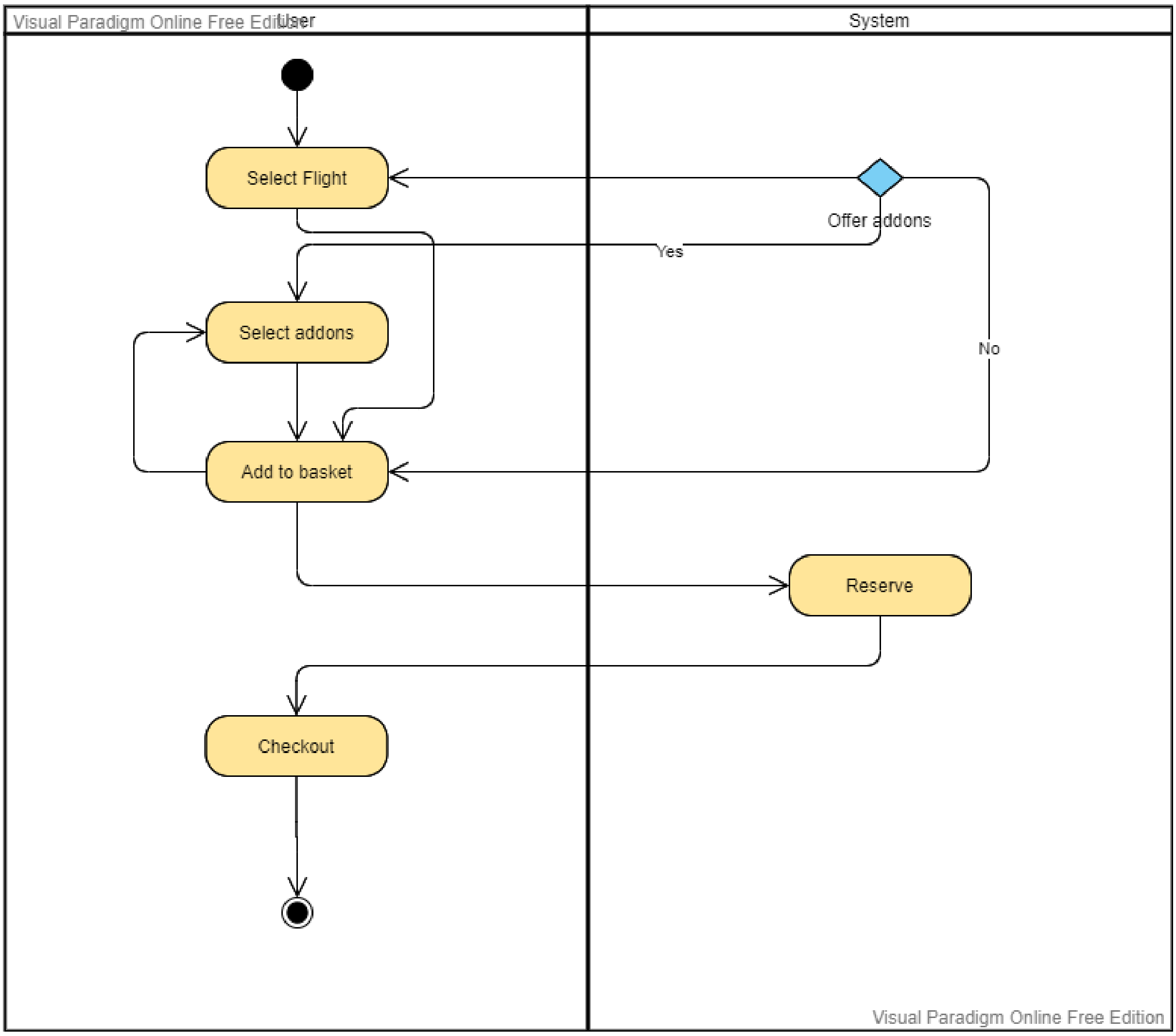
- System failure on the website before step 7 causing a cancellation in their registering of an account

Activity Diagrams:

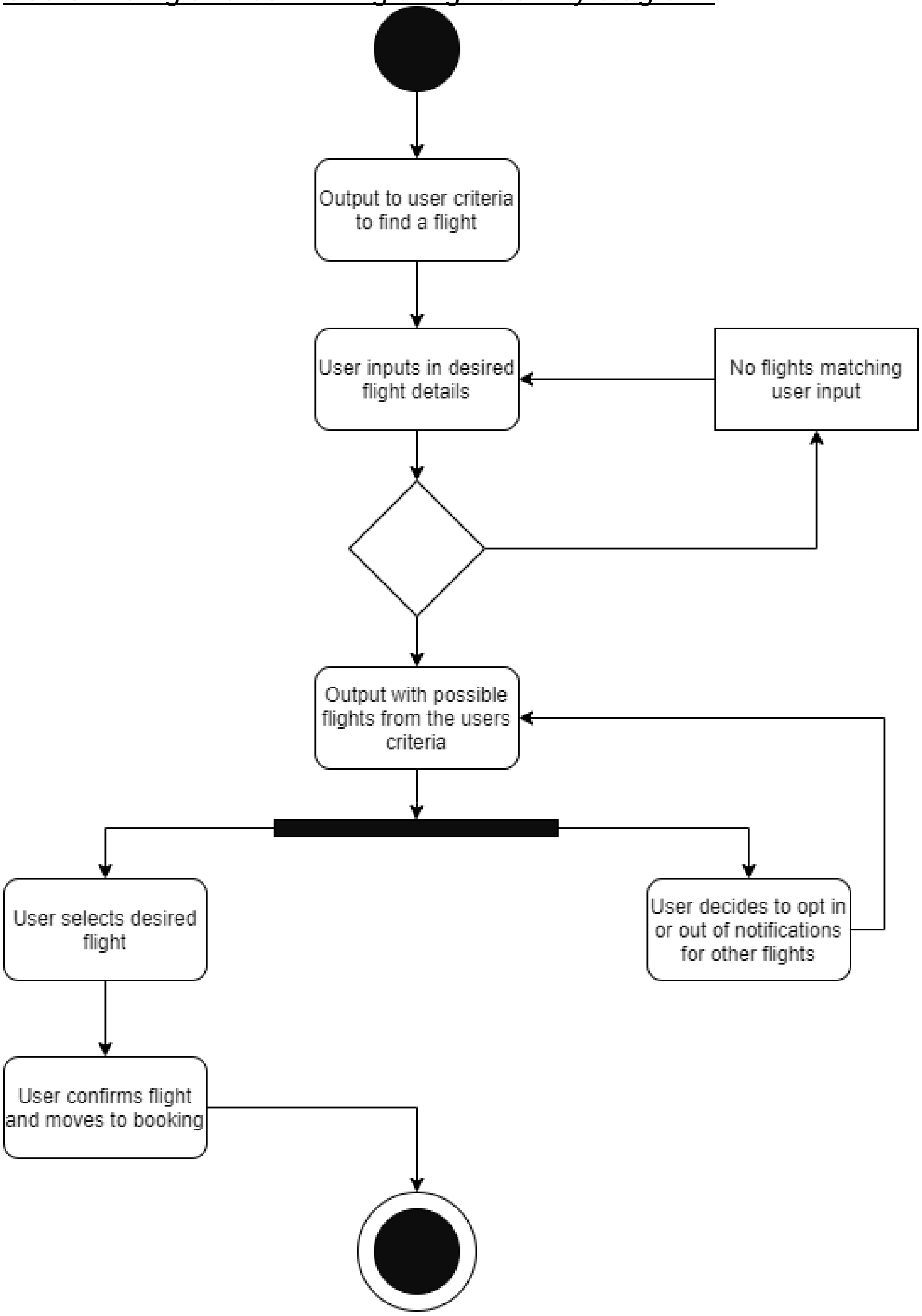
Make a payment activity diagrams:


Visual Paradigm Online Free Edition
Customer
System
Bank
Checkout
Select payment method
Enter Details
Request User Details
Confirm Details
No
Check available funds
Yes
Authorise
Request User to Confirm payment
No
Confirm Payment
Yes
Confirm Flight and add ons
Send Confirmation email and receipt
Visual Paradigm Online Free Edition

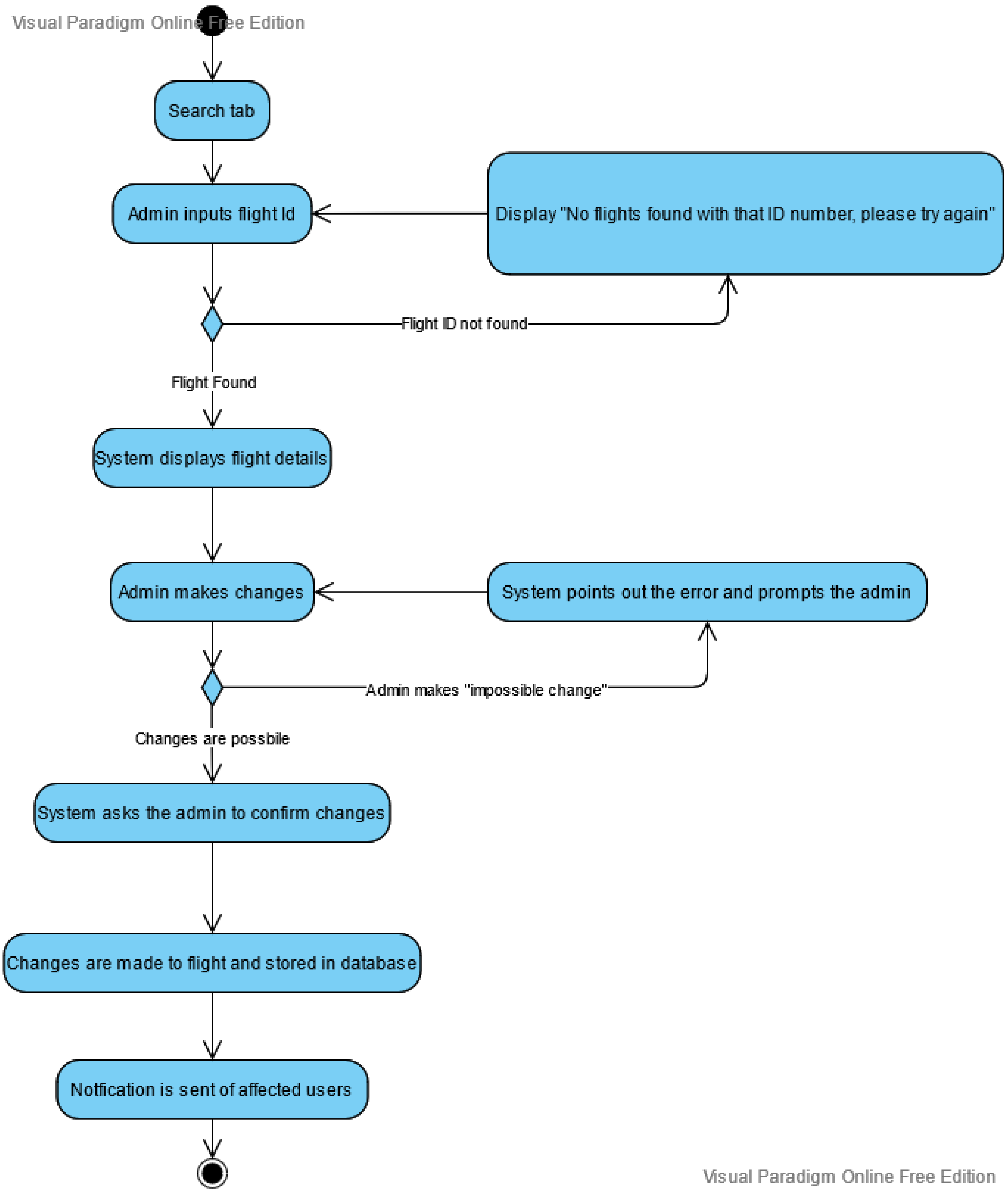
Book a flight Activity Diagram:



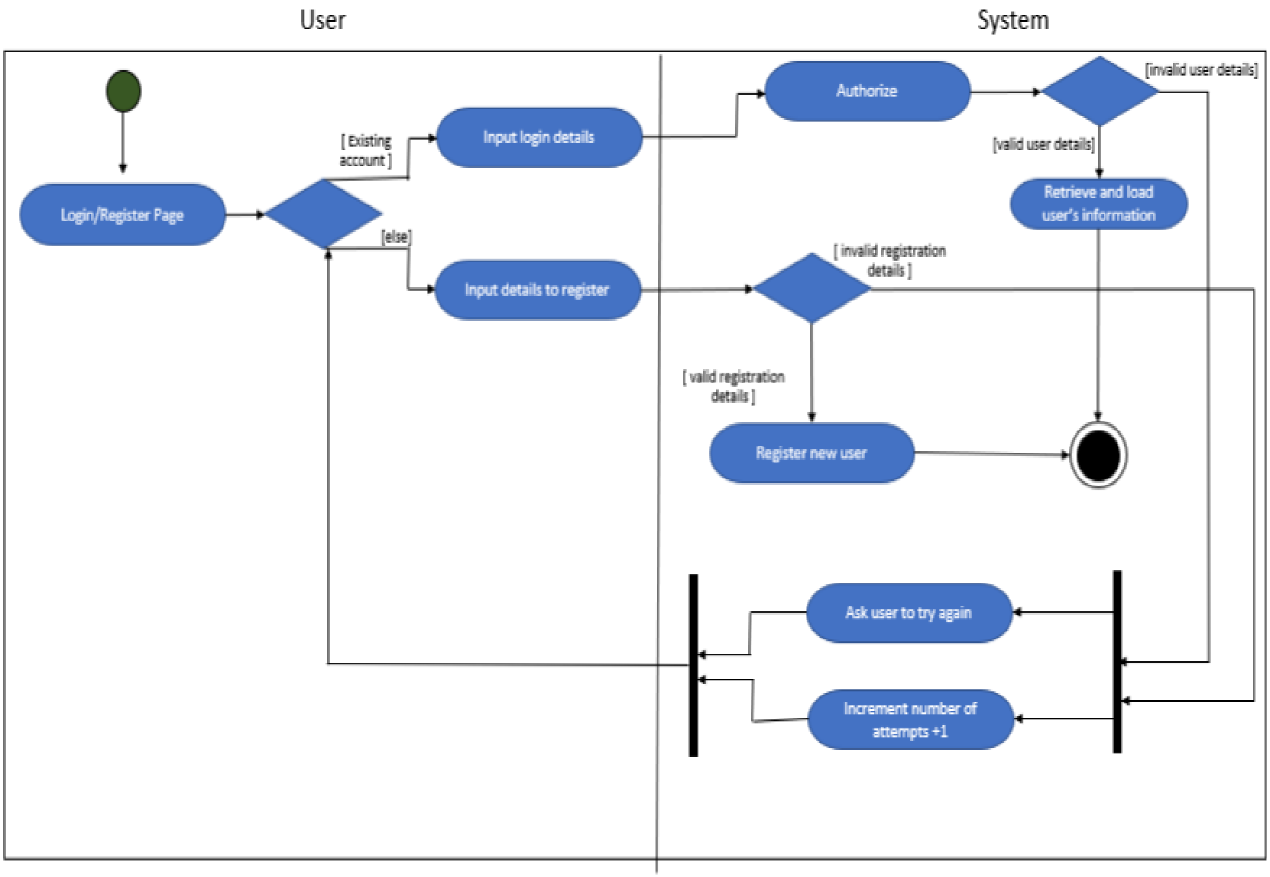
Users finding and confirming a flight activity diagram:


Output to user criteria to find a flight
User inputs in desired flight details
No flights matching user input
Output with possible flights from the users criteria
User selects desired flight
User decides to opt in or out of notifications for other flights
User confirms flight and moves to booking

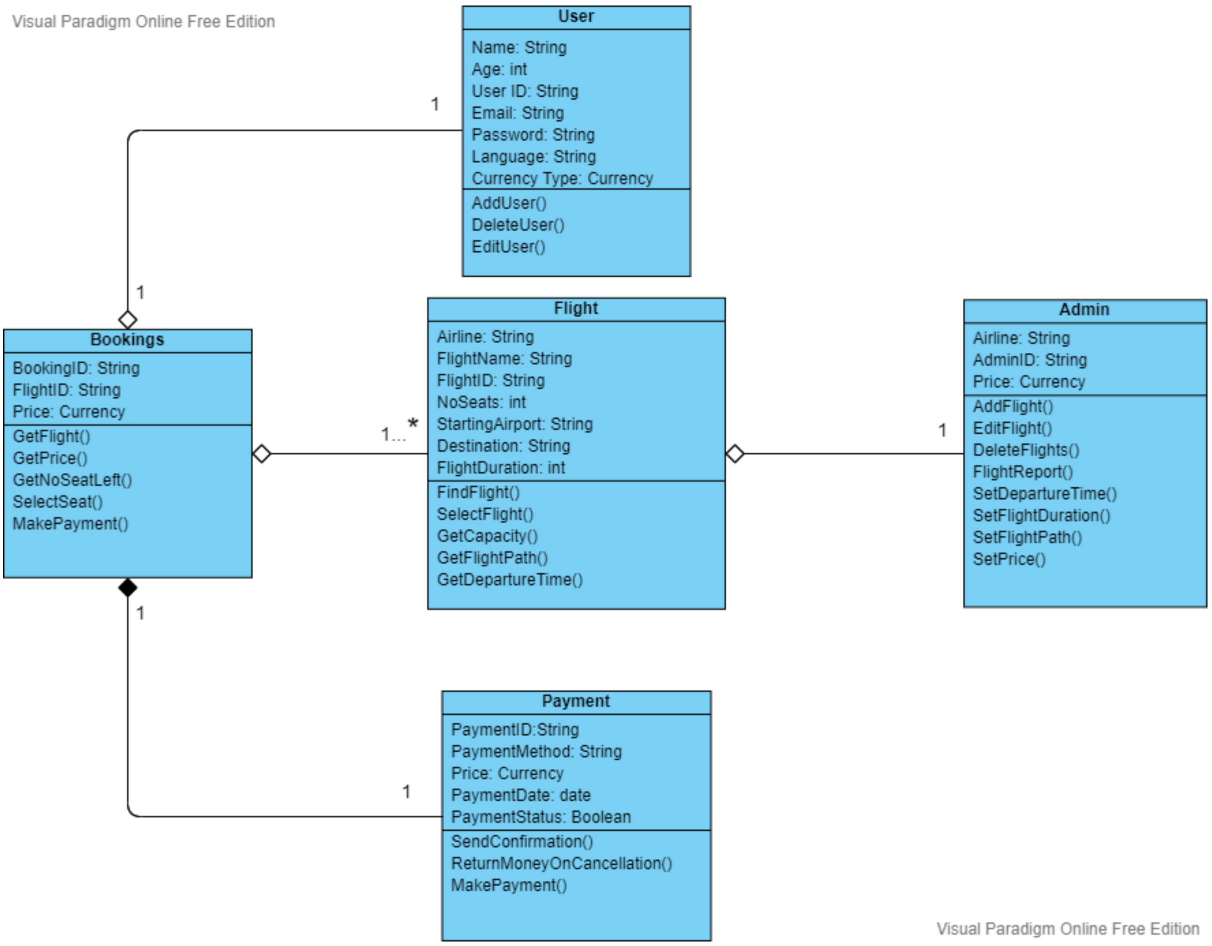
Edit existing flight details diagram:


Visual Paradigm Online Free Edition
Search tab
Admin inputs flight Id
Display "No flights found with that ID number, please try again"
Flight ID not found
Flight Found
System displays flight details
Admin makes changes
System points out the error and prompts the admin
Admin makes "impossible change"
Changes are possbile
System asks the admin to confirm changes
Changes are made to flight and stored in database
Notfication is sent of affected users
Visual Paradigm Online Free Edition

Use Case Login/register diagram:



Class Diagram:


Visual Paradigm Online Free Edition
User
Name: String
Age: int
User ID: String
Email: String
Password: String
Language: String
Currency Type: Currency
AddUser()
DeleteUser()
EditUser()
1
1
Bookings
BookingID: String
FlightID: String
Price: Currency
GetFlight()
GetPrice()
GetNoSeatLeft()
SelectSeat()
MakePayment()
1...*
Flight
Airline: String
FlightName: String
FlightID: String
NoSeats: int
StartingAirport: String
Destination: String
FlightDuration: int
FindFlight()
SelectFlight()
GetCapacity()
GetFlightPath()
GetDepartureTime()
1
Admin
Airline: String
AdminID: String
Price: Currency
AddFlight()
EditFlight()
DeleteFlights()
FlightReport()
SetDepartureTime()
SetFlightDuration()
SetFlightPath()
SetPrice()
1
1
Payment
PaymentID:String
PaymentMethod: String
Price: Currency
PaymentDate: date
PaymentStatus: Boolean
SendConfirmation()
ReturnMoneyOnCancellation()
MakePayment()
Visual Paradigm Online Free Edition