**CHAPTER THREE**

**PROJECT METHODOLOGY**

**3.1** **Chapter Introduction**

The implementation phase of this speech enabled mobile app involved decomposing the whole process into smaller bits and defining the relationship among the constituent bits. Top down coding approach was employed in the application software implementation using Java programming language. This involved dividing the implementation process into subunits or modules and each subunit being further divided into even smaller subs. This process of division is repeated until each unit is sufficiently small enough to be conveniently coded (implemented) from scratch as an independent entity that performs a clearly defined operation.

The analysis and comparison of existing mobile applications was done and this resulted from the review of related implemented methodologies. The critical analysis led to the adoption of the scheme of requirement specifications that highlighted the nature of the airline reservation mobile app implemented in this project work. The process followed in the implementation of the mobile is succinctly written in the following sections of this chapter.

**3.2 Architectural Design**

The technological approach for implementing the speech enabled mobile airline reservation is based on an architecture which provides the necessary framework for the services, components and interfaces. At the back-end, the services include speech authentication and authorization and other services. The hardware and software components that were used provide logical relationships between applications, end-user services, and underlining back-end services. The database, the mobile android phone, and the designed application form the technological infrastructure for accessing, storing and managing flight information. From the diagram in Figure 3.1, when the user request for flight information for booking, the information will be called from the servers, and after the choosing of the necessary flight the request will be forwarded to the server and from there sent to the Payment Plan servers to check for the authenticity of the payment. If the payment is authentic, the request is sent back to the server and server forwarded it to the database of the Airline for successful reservation. After this, the user booking for flight would be alerted that the reservation processing is completed successfully through the phone. However, if the payment is not authentic then there will be no reservation.

In this project, the following services have been defined and implemented System.Passenger, System.Schedule and System.BookedInformation. System.Passenger service exposes Book Flight, Manage Booking and Check In functionalities: System.Schedule service exposes Flight Search and MBRS.BookedInformation service exposes status of a booked flight. Service interface defines actions the end point will take in order to implement the service. Service interfaces Ipassenger, ISchedule and IBookedInformation were defined, then they were implemented in the classes Passenger.cs, Schdule.cs, BookedInformation.cs respectively. Book Flight, Manage Booking, and Check-in are implemented in the service interface Ipassenger.

A service and its clients must use secure communication. At the very least, the transfer of messages from the clients to the service must be secured, and the clients must have a way of authenticating the service.

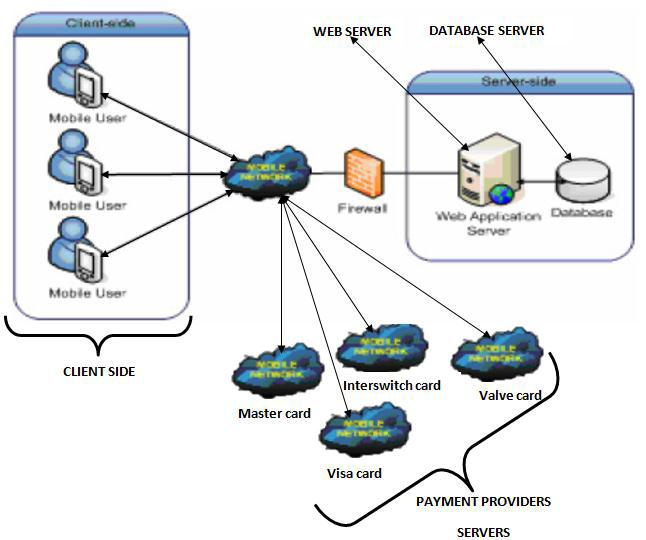


Figure 3.1 Architectural design for the speech enabled reservation system

* 1. **System Requirements**

**3.3.1 Functional Requirements**

A functional requirement defines a function of a software system or its component. The functional requirement specifies specific functionality that defines what a system is supposed to achieve. Each functional requirement has a unique reference ID which is defined as: Functional Requirement-Number.

* 1. FR-1 -The system shall enable users to perform a voice search for flights; something similar to Google voice search.
  2. FR-2-The system shall enable the passenger to book flights.
  3. FR-3-The system shall enable the passenger to manage a booked flight.
  4. FR-4-The system shall enable the passenger to check status of flights.
  5. FR-5-The system shall enable the passenger to check-in online.
  6. FR-6 -The system shall enable users to view information.

The five main scenarios which are identified according to the above functional requirements are described in details as below.

The Scenario 1-Search Flight:

1. The user is permitted to perform a voice search for available flights based on the origin city, destination city, departure date and return date.
2. The system will display any matching records based on the search criteria called.
3. The system will notify the user about the flight availability.
4. If the searched flights are available, then system will display flights which are within a week. Otherwise, the system will prompt to ask the user to re-enter new searching criteria.

The Scenario 2- Book Ticket:

1. The passenger is required to fill in flight search criteria then the system will prompt the passenger to select from the list of available flights.
2. The passenger will select the flight which met his/her schedule then continue booking by filling personal and booking information.
3. The system will generate booking confirmation number upon successful booking.

The Scenario 3-Update Booking:

1. The passenger is required to be logged in using the booking confirmation code before he has privilege to cancel or edit his/her confirmed booking.
2. The passenger will view details of his/her booking.
3. If the passenger wants to edit the booking information he/she will select edit on the booking page then can make changes to the booking.
4. The system will save the changes made by the passenger.
5. If the passenger wants to delete his/her booking he/she will select delete.
6. Once the booking is confirmed to be deleted then the system will delete the data off from the database.

The Scenario 4-Flight Status:

1. The passenger is required to provide the booking confirmation number and flight number of the flight before he has privilege to view status of a given flight.
2. Once the passenger provided this information then the system will display the status of the flight.

The Scenario 5-Check-In:

1. The passenger is required to provide the booking confirmation number and flight number of the flight in order to Check-Into a booked flight.
2. The system will issue a chick in confirmation code in which the passenger use it as boarding pass.

**3.3.2 Non- functional Requirements**

Non-Functional requirements describe user visible aspects of the system that are not designated to the functional behavior of the system. Each non-functional requirement has a unique reference ID which is defined as: Non-functional Requirement-Number. The requirements include from user interface to security issues. Nonfunctional requirements of the system are described as follow:

**3.3.2.1 NFR- 1 Performance**

The Server must respond to user requests maximum in one minute unless the connection is interrupted. The system should also respond to user click maximum in five seconds. Since the system is going to be accessed by different users with different needs, it should be capable of handling and processing their queries quickly. Since the system is an online mobile app system, it is difficult to tell exactly how many users will be using the system at a time. However, the system should handle its users concurrently.

**3.3.2.2 NFR-2 User interface**

Since the system runs on a mobile phone for better management of the small screen, the right type and amount of interactive user interfaces shall be used.

**3.3.2.3 NFR-3 Security**

The system should be developed in a way that the airlines system is exposed for its users in a secured way. The system should be developed in a way by protecting the integrity of data that is exchanged between the system and its users. The system shall also accept only a valid data.

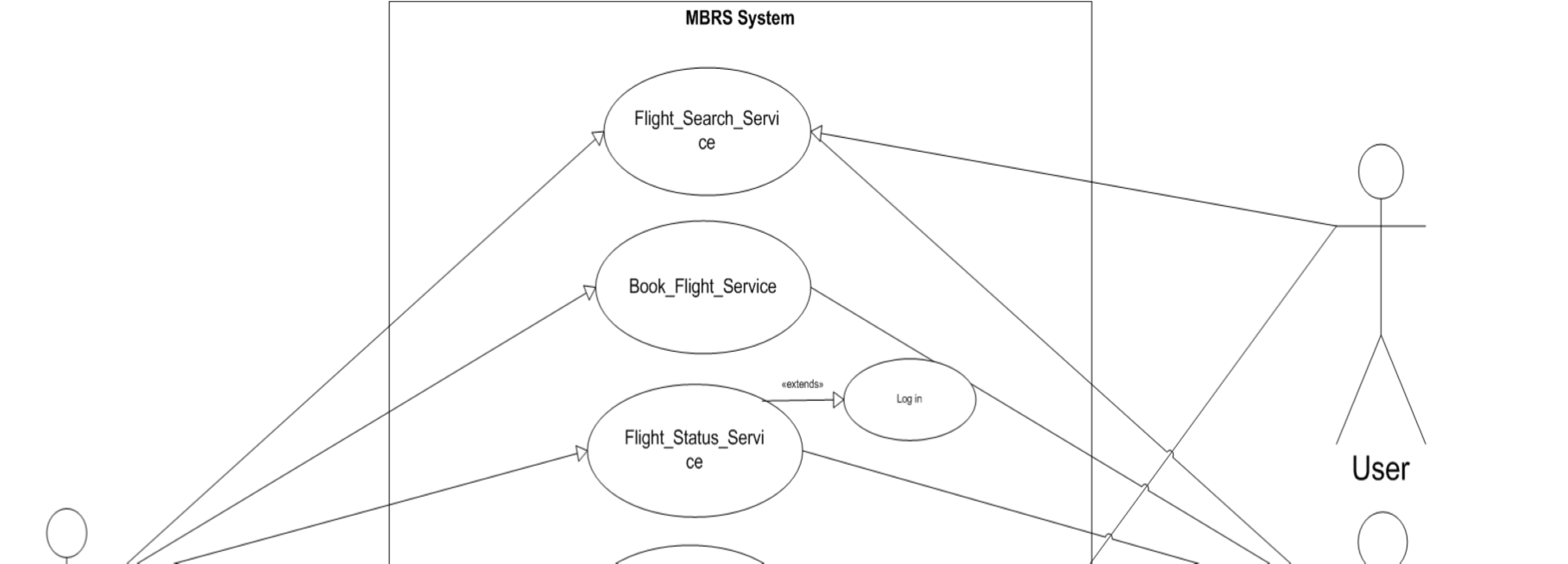
**3.3.2.4 NFR-4 Portability**

As the system primarily aimed to be accessed from mobile phones, which has accesses to internet connection, it should work on mobile devices be it tablets etc., so far they run on android OS.

**3.4 System Analysis Models**

**3.4.1 Use Case Diagram**

The use case diagram represents the functionality (functional requirements) of the system from a user’s point of view. Use cases define the boundaries of the system. The following use cases detail the mobile speech enabled reservation system for Nigeria airline industry.



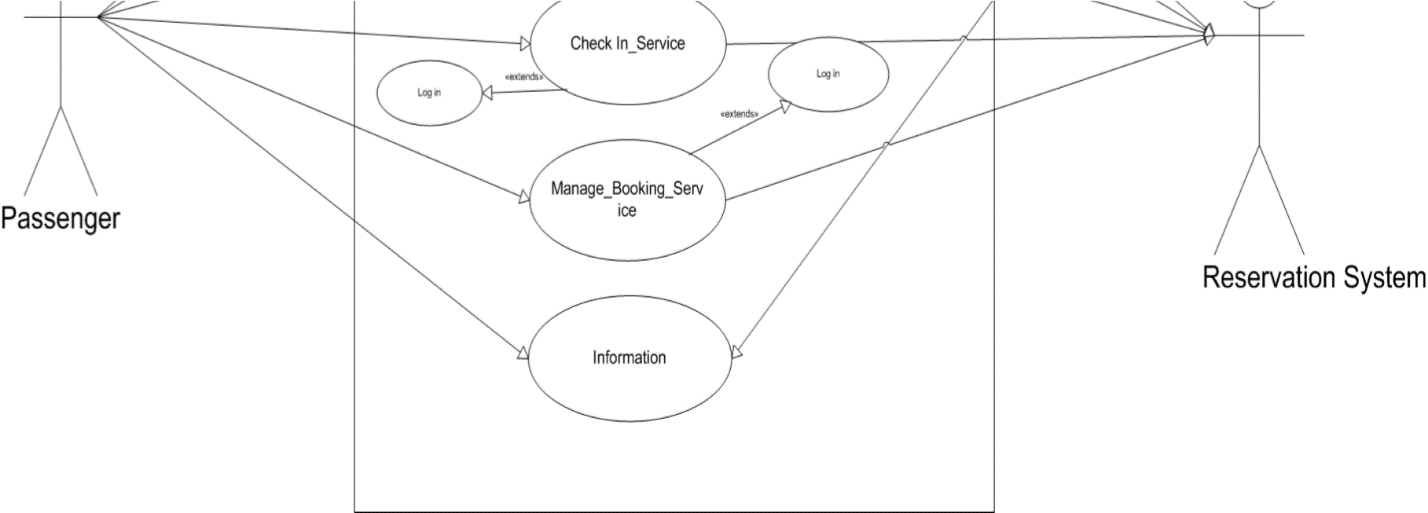


Figure 3.2 Use case diagram for the system

**3.4.2. Use Case Descriptions**

The following tables show the description for the above use case diagram. Each use case has unique identifier UC (use case) number. The description details how the each actor interacts with the system in order to use the services implemented.

**Table 3.1 Use Case Description for Voice (Speech enabled) Search\_Flight\_Service**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **UC 1** | | |
| Use case name | Search Flight | | |
| Actor | User voice | | |
| Description | Allows the user to look for possible flights which met their schedule | | |
| Entry Condition | The user makes the entry of the search by speaking to the mobile phone. | |
| Basic course of action | 1. | The use case begins when the user clicks on Search Flight button and call the search criteria. | |
|  | 2. | The user should provide the origin of the flight. | |
|  | 3. | The user should provide the destination of the flight. | |
|  | 4. | The user should provide the departure date of the requested flight. | |
|  | 5. | The user should say ‘check flight’ to view the available flights. | |
|  | 6. | The information will be validated. | |
|  | 7. | If there is a flight on the given criteria, flights on one week range will be displayed. | |
| Alternative course of action  A | If the user voiced (input) invalid information  6.1. The system displays an error message, prompts the user to correct the problem.  6.2. The user will review the alert details and will be allowed to correct the information again.  If there is no flight based on the user’s criteria.  7.1. The system will notify the user.  7.2. The system will allow the user to perform the voice search again. | | |
| Exit condition | Flights matching the criteria are displayed in list. | | |

**Table 3.2 Use Case Description for Book\_Flight\_Service**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **UC 2** | |
| Use case name | Book flight | |
| Actor | Passenger | |
| Description | Allows the Passenger to Book flights which met their schedule. | |
| Entry Condition | The passenger opens the app on mobile phone | |
| Basic course of action | 1. | The use case begins when the Passenger clicks on the Book flight button to book a flight. |
|  | 2. | The passenger should specify the trip type. |
|  | 3. | The passenger should provide the origin of the flight. |
|  | 4. | The passenger should provide the destination of the flight. |
|  | 5. | The passenger should provide the departure date of the requested flight. |
|  | 6. | The passenger shall provide the return date of the requested flight if it is round trip. |
|  | 7. | The passenger shall specify the number of adult, child and infant passengers. |
|  | 8. | The information will be validated. |
|  | 1. The system will display the available flights. 2. The Passenger will select a flight. 3. The Passenger shall provide personal information. 4. The Passenger shall provide contact information. 5. The Passenger shall provide flight detail. 6. The Passenger shall provide travel document information. 7. The information will be validated. 8. The passenger clicks book. 9. The use case ends when the passenger gets a booking confirmation upon successful booking. | |
| Alternative course of action  A | If passenger enters invalid information.  10.1.The system displays an error message, prompts the user to correct the problem. | |
| If passenger enters invalid information.  15.1. The system displays an error message, prompts the user to correct the problem and allows the passenger to correct the information again. | |
| Exit condition | The booking confirmation number that confirms the flight the passenger books displayed. | |

**Table 3.3 Use Case Description for Flight\_Status\_Service**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **UC 3** | |
| Use case name | Check status | |
| Actor | Passenger | |
| Description | Allows the passenger to view for the status of a flight. | |
| Entry Condition | The passenger opens the app on a mobile phone | |
| Basic course of action | **1.** | The use case begins when the passenger clicks on check status. |
|  | **2.** | The passenger should provide the booking confirmation number of the flight to check the status of a flight. |
|  | **3.** | The passenger clicks check status. |
|  | **4.** | The information will be validated. |
|  | **5.** | The use case ends when the passenger gets status information of the given flight. |
| Alternative course of action  A | If the passenger provide invalid data.  4.1. The system notifies and allows the passenger to correct the information. | |
| Exit condition | The status of flight displayed. | |

**Table 3.4 Use Case Description for Check In\_Service**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **UC 4** | |
| Use case name | Check In | |
| Actor | Passenger | |
| Description | Allows the passenger to check-in online for a flight booked earlier. | |
| Entry condition | The passenger opens the app on a mobile phone | |
| Basic course of action | 1. | The use case begins when the passenger clicks on check in. |
|  | 2. | The passenger should provide the booking confirmation number. |
|  | 3. | The information will be validated. |
|  | 4. | The use case ends when the passenger gets a check in confirmation number. |
| Alternative course of action  A | If the information the passenger entered is invalid.  3.1. The system notifies and allows the passenger to correct the information again. | |
| Exit condition | A check in confirmation number is displayed. | |

**Table 3.5 Use Case Description for Information**

|  |  |
| --- | --- |
| **Identifier** | **UC 5** |
| Use case name | Information |
| Actor | User |
| Description | Allows the user to view basic information. |
| Entry condition | The passenger opens the app on a mobile phone |
| Basic course of action | 1. The user clicks on information. 2. List of information will be displayed. |
| Exit condition | List of information will be displayed. |

**Table 3.6 Use Case Description for Manage\_Booking\_Service**

|  |  |
| --- | --- |
| **Identifier** | **UC 6** |
| Use case name | Manage Booking |
| Actor | Passenger |
| Description | Allows the user to update booking information. |
| Entry condition | The passenger opens browser on a mobile phone and enter the URL. |
| Basic course of action | 1. The user clicks on Manage Booking. 2. The passenger should provide the booking confirmation number. 3. The information will be validated. 4. The passenger will make changes to his/her previous reservation. 5. The use case ends when changes the passenger made are saved. |
| Alternative Course of  Action | If the information the passenger entered is invalid.  3.1. The system notifies and allows the passenger to correct the information again. |
| Exit condition | The use case ends when changes the passenger made are saved. |

**3.4.3 Class Diagram**

A class diagram describes the structure of a system in terms of classes, their attributes, operations (also called methods) and the relationships among the classes. In this section, the identified classes of in the project are presented in Figure 3.3.

**3.4.4 Activity Diagram**

Activity diagram is a graphical representation of work flows of stepwise activities and actions with support for choice, iteration and concurrency. In this section, activity diagrams shown in figures 3.4 to 3.8 are used to describe the work flow of use cases identified in Section 3.4.2.

**3.5 System Design**

The major factor taken into consideration in the design of the new system is the issue of a strong and reliable database for effective form collection and processing, and that is the unit of measurement that is officially used for producing the new system.

**3.5.1 Input Specification and Design**

The word input entails the various data supplied to the system which are processed to give out an output. The input is supplied to the system using computer keyboard mainly as an input device. The major inputs are:

**Travelling Details**

Nature of the trip, departing from, going To, Outbound Cabin/Fare type, Returning, Inbound Cabin/Fare type.

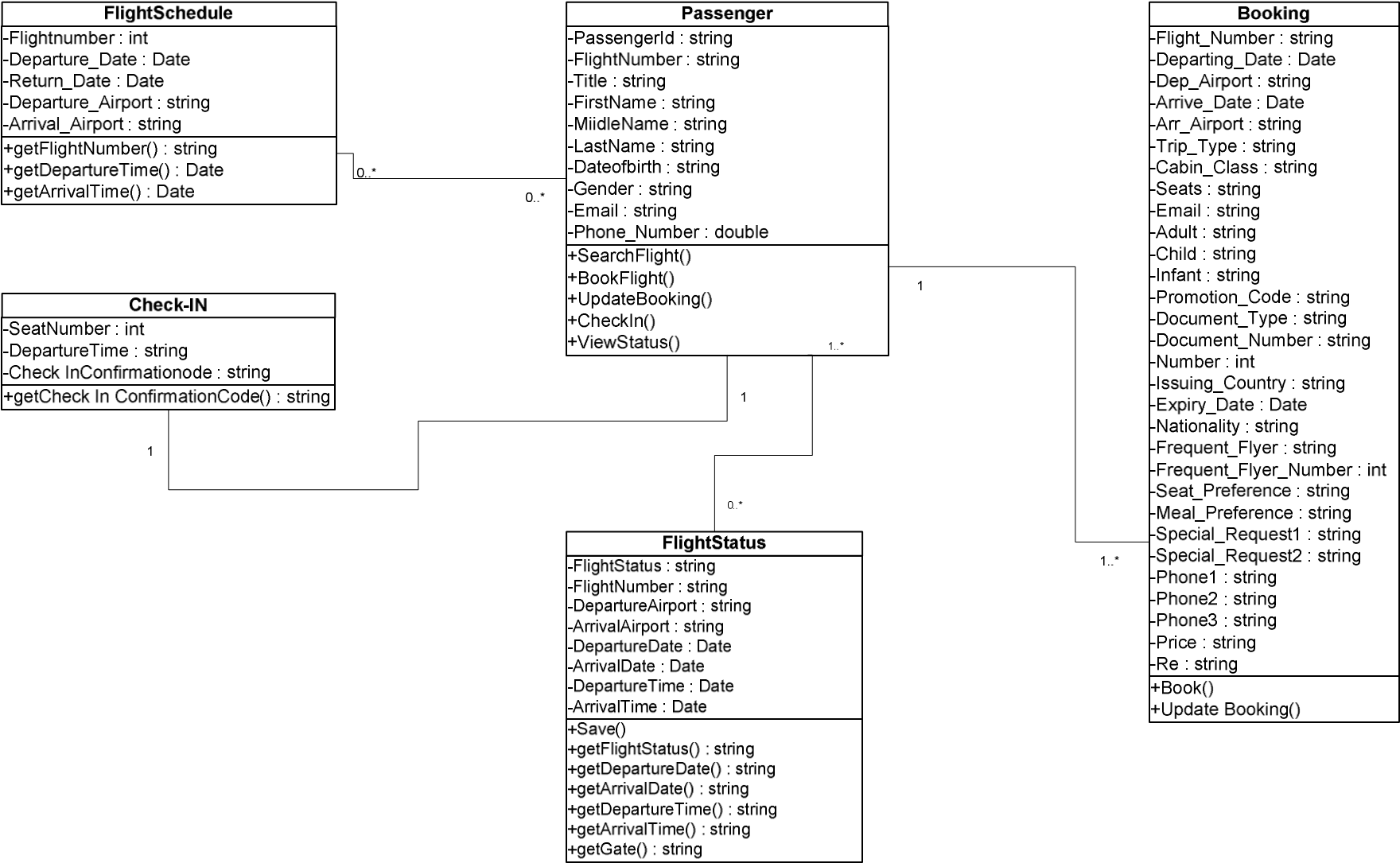


Figure 3.3 Class Diagram

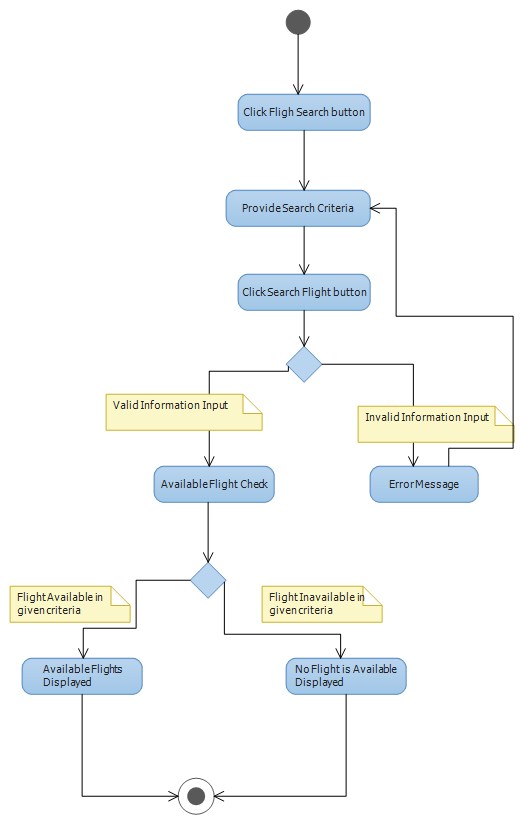


Figure 3.4 Activity diagram for Flight\_Search\_Service

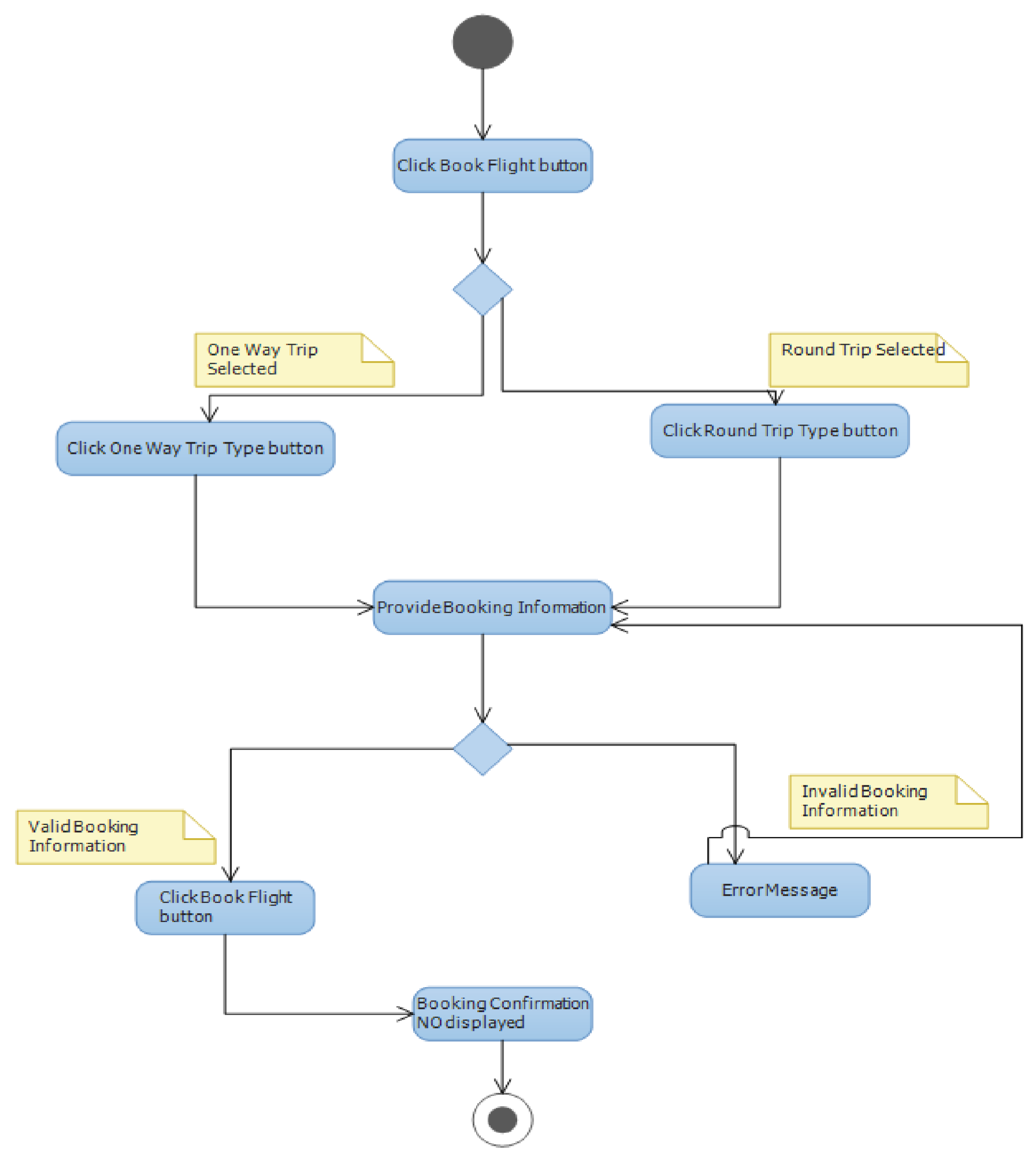


Figure 3.5 Activity Diagram for Book\_Flight\_Service

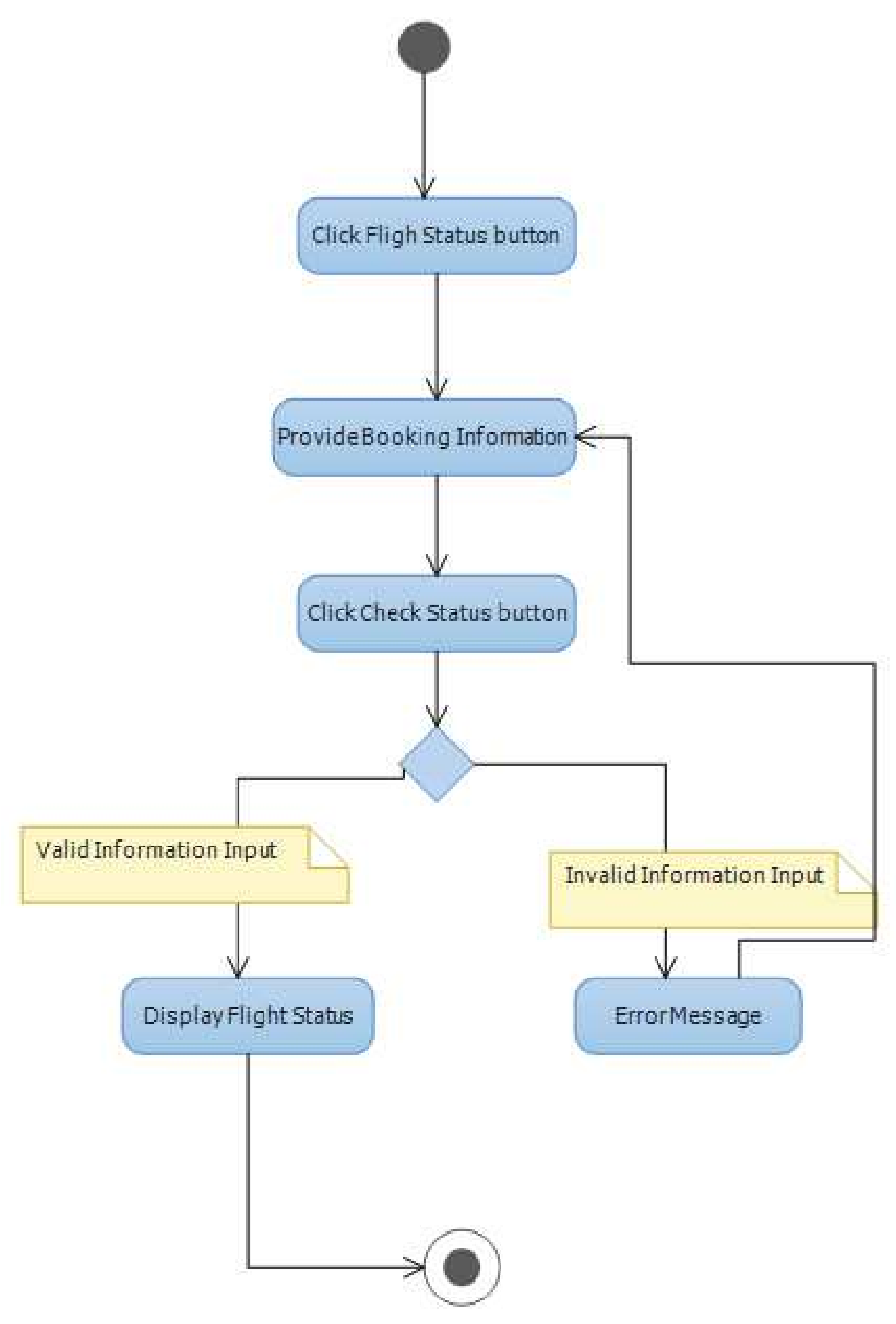


Figure 3.6 Activity Diagram for Flight\_Status\_Service

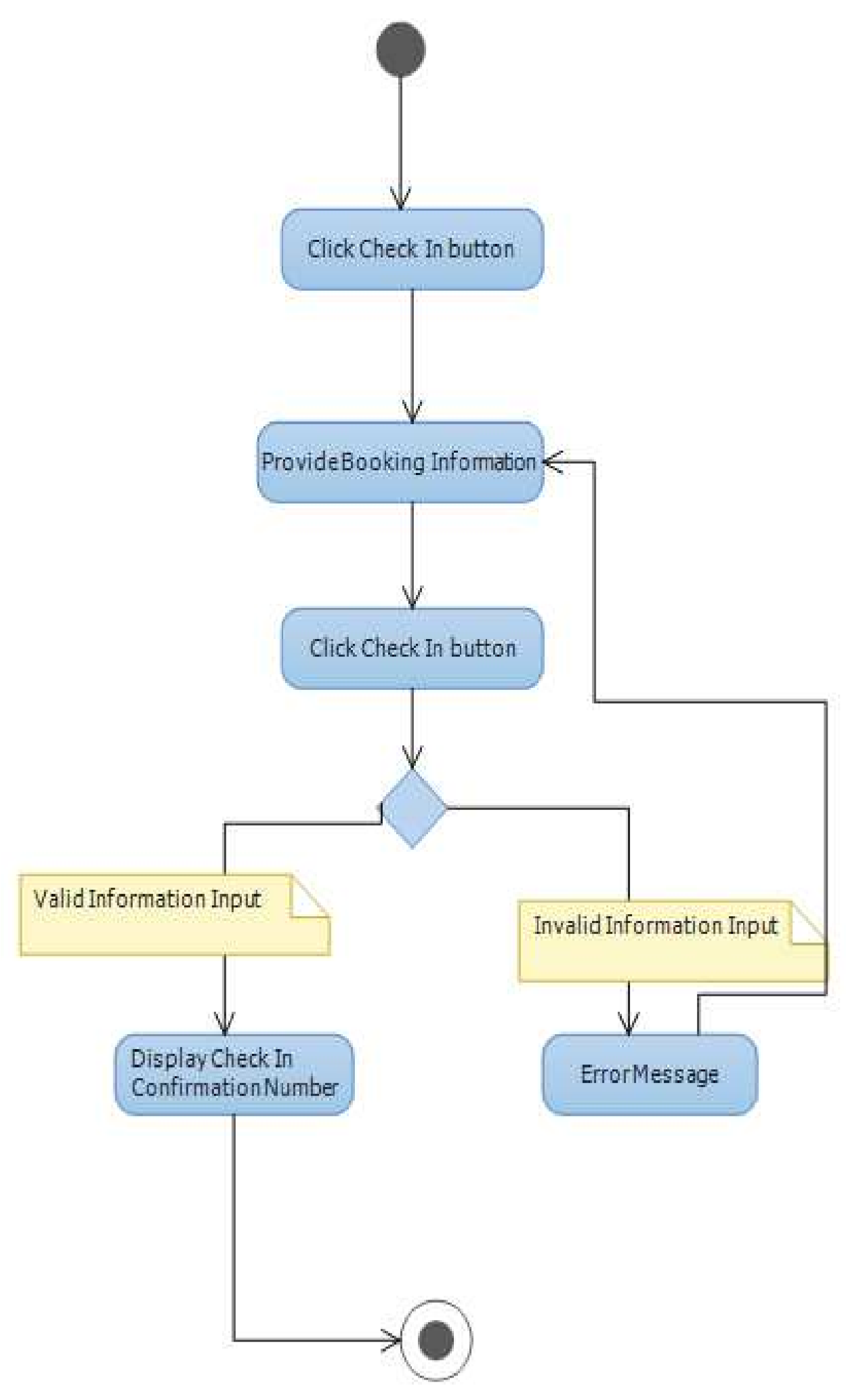


Figure 3.7 Activity Diagram for Check In\_Service

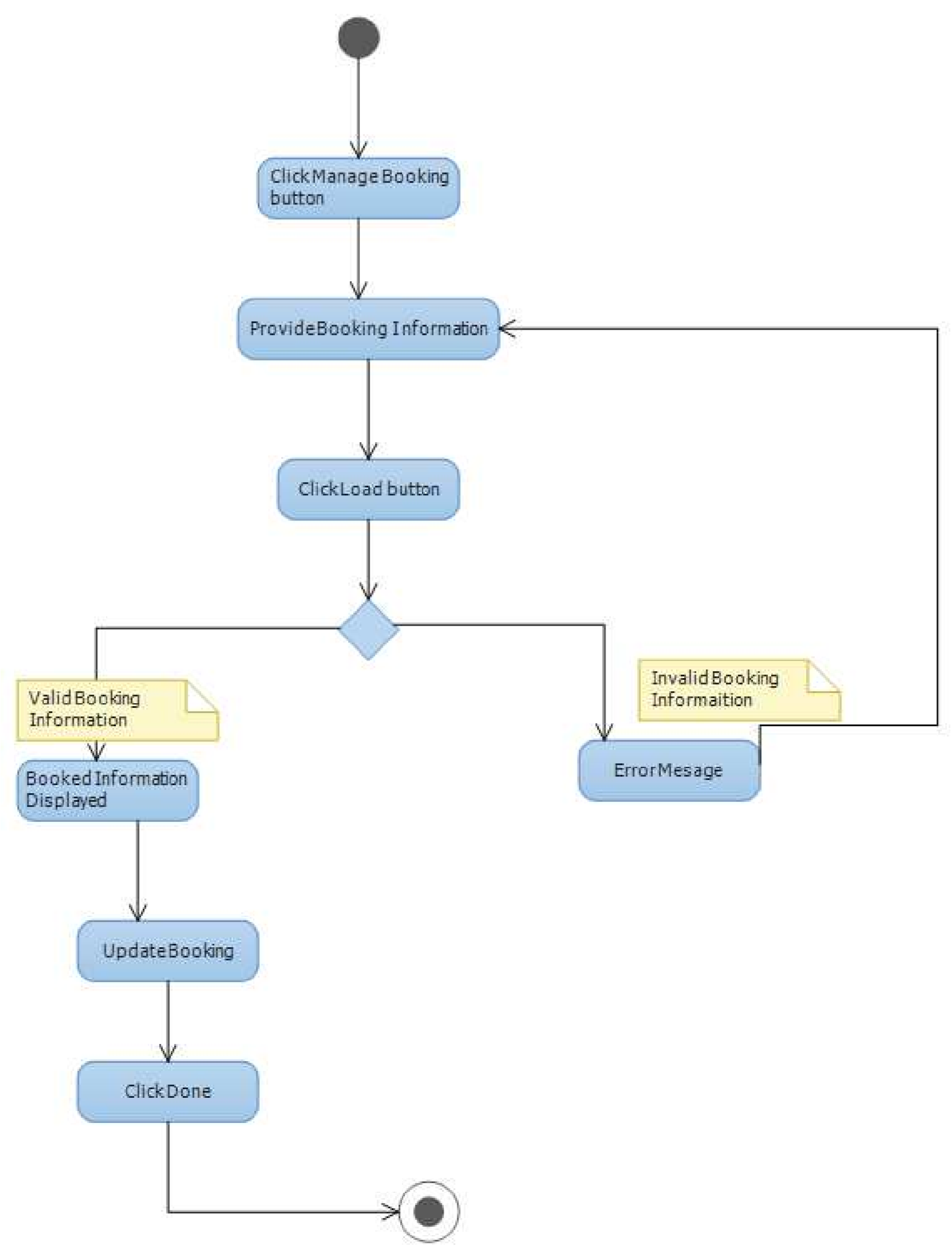


Figure 3.8 Activity Diagram for Manage\_Booking\_Service

**How Many Passengers Travelling**

Adults (Age 12+)

Children (2-11)

Infants (Under 2)

Departing date

Returning date

Currency

**Passenger Details**

Title, First name, last name, mobile phone, contact no, preferred class of flight, seat preference

**Payment Details**

Email address

Confirm email address

Master card no.

**3.5.2 Database Design**

The method of database design chosen for this particular mobile app is the top down file design. This is so that each module could be removed or added without necessarily affecting the entire problem. A database with the following description and properties shown in table 3.7 was used. The design of the new system was carefully drawn on paper considering the old existing systems’ limitations. Flowcharts were used as an effective graphical representation of the program, as well as a design tool and it is aided in the evaluation of a logical correct flow of data and process. Figures 3.10 to 3.12 shows the design flowcharts for the mobile application; consisting of procedure flow, users flow, data entry flowchart, enquiries flowchart, and database management flowchart.

In the database design, the persistent management is presented as it is an abstract and conceptual representation of the data. Figure 3.9 represents the conceptual representation (schema) of the database the project uses. In the diagram, the notation represents entities as boxes, and relationships as lines between the boxes. Different shapes at the ends of these lines represent the cardinality of the relationship (i.e. one or many).

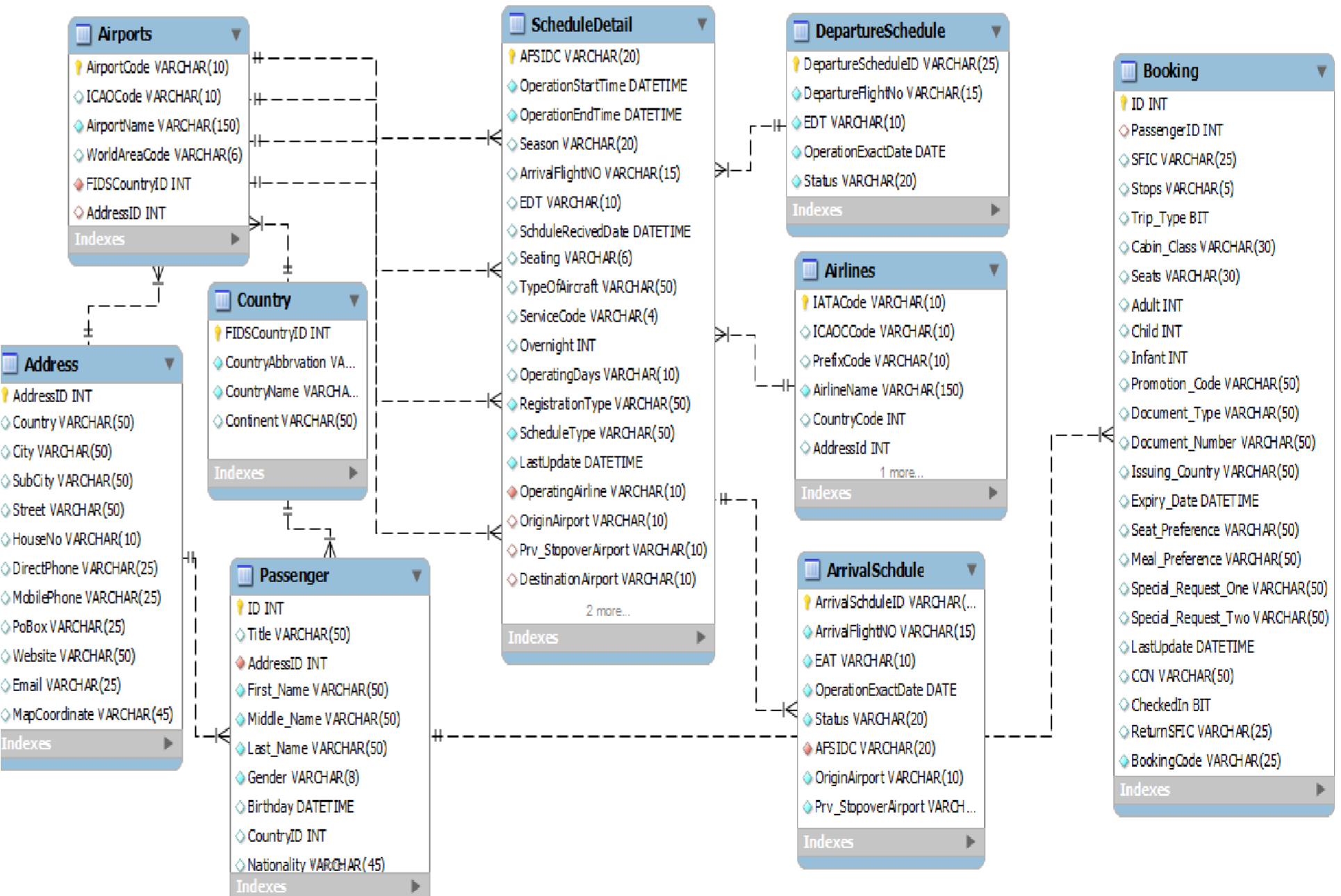


Figure 3.13 Database Schema Diagram

Table 3.7 Database Field Properties

|  |  |  |
| --- | --- | --- |
| **Field name** | **Field type** | **Width** |
| Nature of the trip,  Departing from,  Going to,  Outbound cabin/fare type,  Returning,  Inbound cabin/fare type  Adults (age 12+)  Children (2-11)  Infants (under 2)  Departing date  Returning date  Currency  Title, | Character  Character  Character  Character  Character  Character  Integer  Integer  Integer  Date  Date  Character  Character | 11  11  15  11  11  11  11  3  3  3  10  10  4 |
| First name,  Last name,  Mobile phone,  Contact no,  Flight no,  Seat preference  Email address  Confirm email address  Master card no | Character  Character  Character  Character  Character  Character  Character  Character  Integer | 20  11  35  10  3  25  25  25  16 |

START

ACCEPT INPUT

DATA

CHECT FOR CORRECTION

SELECT MENU OPTION

PROCESSING

GENERATION OF

REPORT

STOP

Figure 3.10 Process Flowchart

START

INPUT MAIN

MENU OPTION

IS

1

=

1

?

IS

=

1

2

?

IS

1

=

3

?

IS

4

=

1

?

WRONG CHOICE TRY AGAIN

DO DATA

ENTRY

DO ENQUIRIES

DO FILE

MAINTENANCE

STOP

Figure 3.11 Input validation flowchart

START

SELECT THE ORDER TO

DISPLAY THE AIRLINES

A

IF

A = 1

IF

A = 2

IF

A = 3

IF

A = 4

DISPLAY

AIRLINES

DISPLAY AIRLINES BY

OFFICERS

STOP

ELSE

DISPLAY AIRLINES BY

GOOD CONTENT

DISPLAY PILOT

DISPLAY ALL AIRLINES

MANIFEST

THEN

Figure 3.12 Enquires Flowchart

START

PRINT “SELECT

CREATE DATA

FILE

IF

A = 1

IF

A = 2

APPEND RECORDS

IF

A = 3

IF

A = 4

ELSE

STOP

UPDATE RECORS

DELETE RECORDS

Figure 3.13 Database Management Flowchart