Faculty of Engineering and Technology
Computer Science Department

Software Engineering – COMP433

First Semester - Fall 2025/2026

Online Flights Booking

Final Report

Group Number: 4

Group Members:

Diaa Badaha	1210478
Nasri Omar	1210810
Omar Shujaiah	1220027
Sameer Ayman	1221561
Ayham Amryah	1221058

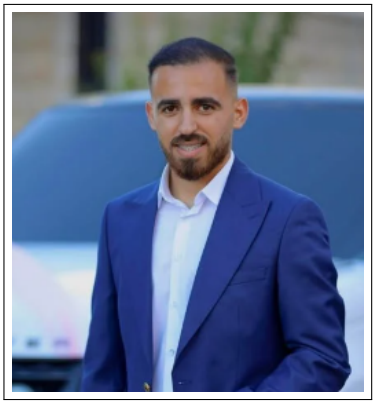
Instructor: Prof. Adel Taweel
Section: 2

Date: Jan 28,2026

Group Members

Diaa Badaha – Project Manager

Nasri Omar – Requirement Engineer



Omar Shujaiah – Secretary

Sameer Ayman – Technical Architect

Ayham Amryah – Programmer

Contents

List of Figures

List of Tables

1 Project Planning and Management

1.1 Writers of the Report

The report was written by:

- Diaa Badaha
- Nasri Omar

1.2 Business Title

Online Flights Booking System

1.3 Group Name

G4

1.4 Name of Students

- Diaa Badaha - 1210478
- Nasri Omar - 1210810
- Omar Shujaiah - 1220027
- Sameer Ayman - 1221561
- Ayham Amryah - 1221058

1.5 Role of Students

- Diaa Badaha – Project Manager
- Nasri Omar – Requirement Engineer
- Omar Shujaiah – Secretary
- Sameer Ayman – Technical Architect
- Ayham Amryah – Programmer

1.6 Project Management Strategy

The project was managed using a structured yet flexible strategy that emphasized clear communication, adaptability, and continuous oversight. Effective communication was maintained throughout the project using Messenger for daily coordination and quick updates, while Discord was primarily used for conducting formal meetings and discussions. This combination allowed the team to balance informal, fast-paced communication with more organized and focused meetings.

Regular meetings were held on a weekly basis, with additional meetings scheduled as deadlines approached or when critical phases required closer coordination. This ensured that progress was continuously monitored and that any issues were addressed promptly before they could escalate.

Task assignment was carried out based on a combination of team members' roles and their availability. This approach helped maintain fairness in workload distribution while ensuring that tasks were handled by the most suitable members. In cases where delays occurred or a member was unavailable, tasks were either reassigned to another member or, when necessary, handled directly by the project manager to maintain progress and meet deadlines.

A hybrid development approach was adopted throughout the project. More structured and well-defined phases followed a Waterfall-like approach, while phases requiring flexibility and iteration were managed using Agile principles. This hybrid strategy allowed the team to adapt to changing requirements and varying task dependencies without compromising structure or clarity.

Decision-making followed a consultative leadership model, where input from team members was actively encouraged and considered. Final decisions, however, were made by the project manager to ensure consistency, efficiency, and alignment with project goals.

For collaborative documentation and design work, the team relied on Google Docs and Overleaf for shared writing and reporting, while Lucidchart was used for creating and refining UML diagrams. These tools enabled real-time collaboration, reduced version conflicts, and ensured that all team members worked with the most up-to-date materials.

1.7 Project Manager Report

This project was both challenging and rewarding, requiring continuous coordination, close monitoring, and active involvement across all phases. One of the main challenges faced was managing scheduling constraints and time pressure, especially as submission deadlines approached. Coordinating team members with different availabilities while maintaining steady progress required careful planning, frequent communication, and flexibility in task reassignment. These challenges were addressed through regular meetings, clear task distribution, and direct intervention when delays or integration issues arose.

As the project manager, my role extended beyond planning and coordination. I was responsible for overseeing progress, reviewing all deliverables, ensuring consistency across sections, and maintaining overall project quality. In several cases, I directly contributed to resolving issues, revising incomplete work, and supporting integration between different components to ensure alignment with project objectives and timelines.

Each team member made valuable contributions to the success of the project. **Nasri Omar** (Requirement Engineer) played a key role in requirements analysis, ensuring clarity, completeness, and alignment with project scope, which reduced ambiguity and rework in later stages. **Omar Shujaiah** (Secretary) contributed through organized documentation, timely task execution, and effective tracking of discussions and decisions. **Sameer Ayman** (Technical Architect) strengthened the system's design by refining architectural decisions and resolving technical ambiguities. **Ayham Amryah** (Programmer) contributed significantly during implementation by addressing technical challenges, refining solutions, and supporting integration efforts.

Overall, I consider the project to be successful. Despite time constraints and coordination challenges, the team demonstrated strong collaboration, responsibility, and adaptability. The project outcome reflects a well-structured system developed through effective communication, balanced workload distribution, and proactive management. This experience closely resembled a real-world software development environment and reinforced the importance of leadership, planning, and teamwork in delivering a high-quality software project.

1.8 Group Members Report

Nasri Omar

Being part of this course and working as the requirements engineer was a rewarding experience. From the beginning, our team collaborated effectively through weekly face-to-face meetings and social media, which helped us share ideas, solve problems, and stay connected. My role involved gathering and documenting system requirements, providing clear direction for the project. Throughout the process, the team remained supportive and committed, even when challenges arose, and we learned a great deal from those experiences. This project also helped me develop an entrepreneurial mindset by improving my understanding of planning, teamwork, and product development, which will be valuable for my future startup. I am truly grateful to Prof. Adel for his guidance and support, and I will always remember this course as a meaningful and unforgettable experience.

Omar Shujaiah

I believe our project was successful in converting real-world business needs into a functional system through effective teamwork and well-structured phases. As the team secretary, I was responsible for organizing and documenting project activities across all phases. I contributed to preparing the business proposal, recording key services and operational details, and maintaining clear documentation of user and system requirements. I also assisted in reviewing system design goals along with the class and component diagrams to ensure consistency and clarity.

Sameer Ayman

Sameer Ayman I believe our project was successful in delivering a clear, well-structured, and scalable design for an online flight booking system. The project effectively translated functional and non-functional requirements into a coherent system architecture, supported by consistent modeling across all phases. As the Technical Architect of the team, I was mainly responsible for designing and validating the overall system structure. I led the component diagrams and overall architecture decisions, ensuring high cohesion and low coupling between system components. I also contributed to defining the system design goals, particularly those related to modularity, reliability, and fault tolerance. In addition, I actively participated in the development and review of the class analysis and detailed class diagrams, as well as reviewing sequence diagrams, to ensure consistency between requirements, design models, and system behavior. Through continuous collaboration with team members, I helped align technical decisions with business requirements, which resulted in a solid and well-documented system design.

Ayham Amryah

As the programmer of the group, I contributed to transforming system requirements and design models into functional and reliable implementations. I worked closely with the team to ensure that the designed use cases and class structures could be effectively translated into executable logic. My role involved implementing core programming components, resolving technical issues, and refining system behavior based on feedback from testing and reviews. Through this project, I strengthened my problem-solving skills and gained valuable experience in aligning code with software engineering documentation. This experience enhanced my understanding of collaborative development and the importance of integrating design and implementation in real-world software projects.

2 Requirement Elicitation, Analysis and Modelling

2.1 Requirement Statement – Business Description

The Online Flight Booking System is designed for a single airline operating at a regional scale, serving both individual and corporate travelers. The airline currently depends on a semi-manual workflow involving staff-operated systems, physical offices, and limited online services. While this approach allows bookings to be completed, it often results in fragmented processes, delayed confirmations, and inefficiencies in managing payments, booking changes, and customer support.

The main objective of the system is to provide a centralized and efficient booking platform that improves customer experience and operational effectiveness. Customers can search for available flights based on departure location, destination, travel date, and number of passengers, viewing detailed information such as schedules, seat classes (Economy, Business, VIP), availability, and prices. Although flight browsing is available to all users, only registered users are permitted to create and manage bookings, ensuring secure handling of customer data and transactions.

The system supports flexible booking workflows, enabling customers to enter passenger details (full name, date of birth, email, passport/ID), select seat classes, add optional services such as baggage, meals, or special assistance, and choose between immediate online card payment or a pay-later option. Pay-later bookings can be completed at airline offices with staff confirming cash payments through the system. The system ensures booking transactions are atomic to prevent partial bookings or data inconsistencies.

Post-booking management is a key feature of the system. Customers can access booking details using their booking reference number and passenger last name, download e-tickets, track payment status, and review applicable cancellation policies and fare rules. Bookings may be modified or canceled according to airline policies, including special handling for critical cases such as medical or family emergencies requiring proof document uploads, which are managed through controlled workflows with booking locking mechanisms to prevent concurrent modifications.

Authorized staff members support system operations by managing bookings on behalf of customers, confirming payments, processing cancellations, issuing tickets, and updating records. All staff actions are logged and trigger automatic notifications to customers via email and SMS, ensuring transparency throughout the booking lifecycle.

Customer support is integrated through a dedicated interface allowing customers to submit support tickets, receive unique ticket identifiers, and track request status. System requirements were derived from airline operational needs and customer expectations, focusing on high cohesion and low coupling in design, reliability through fault-tolerant mechanisms, and flexibility while managing the complete booking lifecycle efficiently.

2.2 User and System Requirements

2.2.1 User Requirements (MODIFIED)

[Leader: Nasri, Reviewing: Diaa, Discussing: Sameer, Finalizing: Ayham, Re-drawing: Omar]

- **UR0: User Authentication** The system shall support user authentication with two user types: Customer and Staff.
- **UR1: Search for Available Flights** The system shall enable customers to search for available flights by specifying departure location, destination, travel date, and number of passengers.
- **UR2: Book a Flight** The system shall enable customers to book a selected flight by providing passenger information, selecting seat class and optional services, and completing payment.
- **UR3: Modify an Existing Booking** The system shall enable customers to modify their confirmed bookings by changing flight dates, adjusting seat selections, or updating optional services according to airline modification policies.
- **UR4: Cancel a Booking** The system shall enable customers to cancel their confirmed bookings and process refunds according to applicable cancellation policies.
- **UR5: Complete Pending Payment** The system shall enable customers to complete payment for pending bookings by paying cash or card at airline offices or authorized agents, with staff confirmation.
- **UR6: View Booking Details** The system shall enable customers to retrieve and view their booking details, including flight information, passenger information, payment status, and e-ticket, using their booking reference number.
- **UR7: Manage Customer Bookings (Staff Function)** The system shall enable staff members to manage customer bookings on behalf of customers when requested or required.
- **UR8: Submit and Track Support Requests** The system shall enable customers to submit support requests or inquiries through a dedicated support form and track the status of their submitted requests using a unique ticket identifier.

2.2.2 System Requirements (MODIFIED)

[Leader: Nasri, Reviewing: Diaa, Discussing: Omar, Finalizing: Sameer, Re-drawing: Ayham]

UR0: User Authentication

- **SR0.1** The system shall support two user types: Customer and Staff (airline employees).
- **SR0.2** The system shall allow new customers to register by providing name, email, and password.
- **SR0.3** Staff accounts shall not be self-registered and shall be provisioned by the organization.
- **SR0.4** Users shall log in using email and password to access the system.
- **SR0.5** Customers must be logged in to create bookings, and bookings shall be automatically linked to their accounts.

UR1: Search for Available Flights

- **SR1.1** The system shall provide a search form requiring departure location, destination location, departure date, and number of passengers.
- **SR1.2** The system shall validate that departure and destination locations are different.
- **SR1.3** The system shall validate that the departure date is not earlier than the current date.
- **SR1.4** The system shall retrieve flights matching the search criteria.
- **SR1.5** The system shall retrieve for each flight: flight number, airline, departure and arrival times, available seat classes (Economy, Business, VIP), price per class, and seat availability.
- **SR1.6** The system shall return a "No flights available" message when no matches are found.
- **SR1.7** The system shall enable customers to filter results by price range, departure time, or airline.
- **SR1.8** The system shall return search results within 5 seconds for 95% of search requests.

UR2: Book a Flight

- **SR2.1** The system shall require passenger information including full name, date of birth, email address, and passport/ID number.
- **SR2.2** The system shall validate that passenger information is complete and correctly formatted.
- **SR2.3** The system shall verify that passengers under 18 are accompanied by an adult in the booking.
- **SR2.4** The system shall enable seat class selection based on availability.
- **SR2.5** The system shall enable selection of optional services such as baggage, meals, and special assistance.
- **SR2.6** The system shall calculate the total price including base fare, taxes, and optional services.
- **SR2.7** The system shall allow customers to choose between online card payment and pay-later option.
- **SR2.8** For online payments, the system shall require cardholder name, card number, expiry date, and CVV.
- **SR2.9** The system shall process payment and receive confirmation or rejection.
- **SR2.10** The system shall create a booking with status "Confirmed" or "Pending Payment".
- **SR2.11** The system shall generate a unique booking reference number.
- **SR2.12** The system shall generate an e-ticket for confirmed bookings.
- **SR2.13** The system shall send booking confirmation and e-ticket to the customer's email.
- **SR2.14** The system shall reserve seats for pending bookings for 48 hours.
- **SR2.16** The system shall ensure booking transactions are atomic to prevent data inconsistencies.

UR3: Modify an Existing Booking

- **SR3.1** The system shall retrieve bookings using booking reference number and passenger last name.
- **SR3.2** The system shall display current booking details and applicable modification policy.
- **SR3.3** The system shall allow selection of modification type.
- **SR3.4** The system shall allow customers to indicate critical cases.
- **SR3.5** The system shall require proof documents for critical cases.
- **SR3.6** The system shall validate document format and size.
- **SR3.7** The system shall lock the booking during modification.
- **SR3.8** The system shall suggest alternative flights.
- **SR3.9** The system shall verify seat availability.
- **SR3.10** The system shall calculate modification fees.
- **SR3.11** The system shall apply fee waivers for approved critical cases.
- **SR3.12** The system shall display modification summary.
- **SR3.13** The system shall process additional payments if required.
- **SR3.14** The system shall update booking and seat inventory.
- **SR3.15** The system shall unlock the booking after completion.
- **SR3.16** The system shall generate and send updated e-ticket.
- **SR3.17** The system shall prevent modification outside allowed window.

UR4: Cancel a Booking

- **SR4.1** The system shall retrieve booking details using reference and passenger name.
- **SR4.2** The system shall retrieve applicable cancellation policy.
- **SR4.3** The system shall calculate refundable amount.
- **SR4.4** The system shall display cancellation summary.

- **SR4.5** The system shall update booking status to "Cancelled".
- **SR4.6** The system shall release reserved seats.
- **SR4.7** The system shall initiate refund if applicable.
- **SR4.8** The system shall send cancellation confirmation email.

UR5: Complete Pending Payment

- **SR5.1** The system shall allow staff to retrieve pending bookings.
- **SR5.2** The system shall display payment amount and deadline.
- **SR5.3** The system shall verify booking is not expired.
- **SR5.4** The system shall allow staff to select payment method.
- **SR5.5** The system shall confirm payment.
- **SR5.6** The system shall update booking status to "Confirmed".
- **SR5.7** The system shall create a payment record.
- **SR5.8** The system shall generate e-ticket and receipt.
- **SR5.9** The system shall notify customer by email.

UR6: View Booking Details

- **SR6.1** The system shall require booking reference and passenger last name.
- **SR6.2** The system shall validate booking credentials.
- **SR6.3** The system shall retrieve complete booking details.
- **SR6.4** The system shall provide e-ticket download for confirmed bookings.
- **SR6.5** The system shall display payment deadline for pending bookings.
- **SR6.6** The system shall display "Booking not found" if no match exists.

UR7: Manage Customer Bookings

- **SR7.1** The system shall allow staff to search bookings using multiple identifiers.
- **SR7.2** The system shall retrieve full booking details.
- **SR7.3** The system shall allow staff to modify, cancel, confirm payments, and reissue tickets.
- **SR7.4** The system shall notify customers of staff-initiated changes.

UR8: Submit and Track Support Requests

- **SR8.1** The system shall provide a support request form.
- **SR8.2** The system shall validate mandatory fields.
- **SR8.3** The system shall generate unique support ticket IDs.
- **SR8.4** The system shall store support request details.
- **SR8.5** The system shall send confirmation email.
- **SR8.6** The system shall allow customers to track ticket status.
- **SR8.7** The system shall notify customers of ticket updates.

2.3 Scenarios

2.3.1 Scenario #1: Book a Flight Ticket Online [Diaa]

Initial Assumption

The customer is a registered user of the airline online booking system and is successfully logged in. The system is operational, connected to the flight database and payment services, and there are scheduled flights available for booking. The customer has access to the internet and possesses valid personal and travel information required for booking.

Normal

The customer books a flight ticket online through the airline booking system. The customer searches for available flights by entering the departure city, destination, travel date, and number of passengers. The system retrieves and displays a list of available flights along with schedules, prices, and ticket classes. The customer selects a suitable flight, chooses a ticket class (Economy, Business, or VIP), and proceeds to enter passenger details such as name, identification information, and contact details. The customer then selects optional services such as seat selection and baggage options. After reviewing the booking summary, the customer chooses to pay online using a valid card. The system securely processes the payment through the bank gateway, confirms the transaction, finalizes the booking, generates a reservation reference number, and issues an electronic ticket (e-ticket) to the customer via the system and email. The booking is stored in the system database successfully.

Alternative

The customer books a flight ticket online but chooses a different business process to complete the booking. After selecting the flight, ticket class, and entering passenger details, the customer selects the "pay later / cash payment" option instead of online card payment. The system temporarily reserves the selected seats and generates a booking reference with a pending payment status. The customer is instructed to visit the airline office or authorized agent within a specified time period to complete the payment in cash. Once the payment is confirmed by the staff, the system updates the booking status to confirmed and issues the electronic ticket to the customer. The reservation is successfully completed without online payment.

Error

The customer attempts to book a flight ticket online and proceeds to the payment stage using a card. After entering valid card details, the system sends the payment request to the bank authorization service, but the transaction is declined due to insufficient funds or bank authorization failure. The system displays an error message indicating that the payment could not be completed, and the booking is not finalized. The selected seats are released after a timeout period, and no reservation reference or ticket is issued.

Error

The customer books a flight ticket online and completes all booking steps successfully, but during the final confirmation stage, the system fails to save the booking due to a database or system connectivity error. The system notifies the customer that the booking process could not be completed and advises them to retry later or contact customer support. No ticket is issued, and the transaction is rolled back to prevent inconsistent data.

2.3.2 Scenario #2: Change Flight Date [Nasri]

Initial Assumption

A registered customer is logged into the flight booking system and already has a confirmed, paid booking with a valid reservation reference (PNR). The ticket is within the allowed change window and its fare rules allow date changes (or allow changes only in critical cases). The customer has a critical reason (medical emergency) and can provide supporting documentation if required by policy. The system is online and connected to the airline inventory and payment gateway services.

Normal

The customer opens "My Bookings", selects the confirmed reservation, and clicks "Change Flight Date." The system first displays the ticket's change policy (change window, possible fees, and fare difference rules) and asks the customer to select a critical-case reason and upload proof (medical report) if required. After the customer submits the reason, the system validates eligibility and temporarily locks the booking to prevent parallel modifications. The customer selects a new travel date, and the system lists available flights for the same route showing departure/arrival time, remaining seats, and available classes (VIP/Business/Economy), along with baggage and seat options. The customer chooses a replacement flight in the same class, and the system re-checks availability in real time, validates fare rules, and calculates any change fee plus the fare difference. A clear summary is shown (old itinerary, new itinerary, fees, total difference, and refund/charge direction). The customer pays the difference by card, and once the payment is authorized successfully, the system commits the change: it releases the old seat inventory, reserves the new seat, updates the booking record, and issues an updated e-ticket/itinerary. Finally, the system sends confirmation on-screen and by email/SMS, and updates the financial records (invoice/receipt) to reflect the additional payment and the modification history for audit purposes.

Alternative

Instead of changing the date online, the customer contacts customer support (or visits the agency office) and provides the reservation reference and identification details. The staff verifies the customer identity, checks the booking's fare rules and the airline's critical-case policy, and requests the necessary supporting proof if it was not already provided. The staff searches for alternative flights that match the route and the customer preferences (time, class, baggage needs, special services) and proposes one or more valid options. After the customer confirms the preferred option, the staff informs the customer of the exact change fee and fare difference and collects payment using an offline method such as cash at the office or bank transfer confirmation. Once payment is confirmed, the staff completes the modification in the system, which updates the reservation, reserves the new seat, issues the updated e-ticket, and sends the new itinerary to the customer, while

ensuring all financial records and modification logs are updated correctly.

Error

The customer selects a new flight date and the system calculates the change fee and fare difference. The customer proceeds to pay the required amount by Visa/card, but the bank/payment gateway rejects the transaction (for example, insufficient balance, incorrect card verification, or gateway timeout). As a result, the system does not confirm the modification and keeps the original booking unchanged. The system displays a clear message that payment failed, records the failed attempt, and offers the customer realistic next actions such as retrying the payment, using another payment method, or contacting customer support to complete the change through an alternative process.

Error

The customer selects the new flight and successfully completes payment for the fare difference. After payment confirmation, the system attempts to update the reservation and reserve the new seat in the airline inventory, but a system failure occurs (for example, failure to retrieve/update booking data from the database, or temporary unavailability of the airline inventory service). The modification is not completed, and the system prevents inconsistent results by keeping the original booking active and marking the change request as "failed" or "pending verification" based on policy. The system notifies the customer that the change could not be finalized due to a technical issue, provides a support reference number, and automatically alerts customer support/finance to verify the booking status and ensure the customer is not charged without receiving an updated ticket.

2.3.3 Scenario #3: Complete Pending Booking (Pay Later / Cash Payment at Office) [Omar]

Initial Assumption

The customer is a registered user of the airline online booking system and is successfully logged in. The customer has already created a booking using the "Pay Later / Cash Payment" option, so the booking exists in the system with status Pending Payment and has a valid reservation reference (PNR). The pending booking is still within the allowed payment time window (before it expires). The system is operational and connected to the booking database. The airline office/authorized agent staff can access the system to confirm cash payments and update booking status. The customer has valid identification and booking information.

Normal

The customer completes a pending booking through cash payment at the airline office or authorized agent. The customer visits the office and provides the reservation reference (PNR) and identification details. The staff retrieves the pending booking in the system and displays booking details including flight information, ticket class, passenger details, total amount due, and the payment deadline. The staff verifies the customer identity and confirms that the booking is still valid and not expired. The customer pays the required amount in cash. The staff records the cash payment in the system and confirms the payment. The system updates the booking status from Pending Payment to Confirmed, issues the electronic ticket (e-ticket), generates the final invoice/receipt, and stores the completed booking in the system database. The system sends booking confirmation and the e-ticket to the customer via the system and email/SMS. The reservation is successfully completed.

Alternative

The customer completes the pending booking but follows an alternative confirmation process. Instead of paying directly at the office, the customer pays through bank transfer (or presents a transfer receipt) and provides the proof to the staff. The staff retrieves the pending booking and reviews the transfer proof. After verifying the transfer details (amount, reference, date) and confirming it with the finance/manager account if required, the staff marks the payment as confirmed in the system. The system updates the booking status to Confirmed, issues the e-ticket, generates the final receipt, and sends confirmation to the customer. The booking is completed successfully without direct cash payment at the counter.

Error

The customer attempts to complete the pending booking, but the booking has expired because the payment deadline has passed. The staff searches the PNR and the system shows that the booking status is Expired/Cancelled and the temporary seat hold has

been released. The system displays an error message indicating that the booking can no longer be completed. No payment is accepted, no ticket is issued, and the customer is instructed to create a new booking using the online system.

Error

The customer pays in cash and the staff attempts to confirm the payment, but the system fails to update the booking due to a database or connectivity error. The system notifies the staff that the confirmation process could not be completed and prevents issuing an e-ticket to avoid inconsistent data. The staff provides the customer with a manual payment receipt/reference and advises them to retry shortly or contact support. The system logs the failure for follow-up, and the booking remains Pending Payment until the issue is resolved (or the transaction is verified and completed).

2.3.4 Scenario #4: Search for Available Flights [Sameer]

Initial Assumption

The customer is a registered user of the airline online booking system and has access to the system through the internet. The system is operational, connected to the flight database, and contains scheduled flights with up-to-date schedules and prices. The customer has the necessary travel information such as departure city, destination, and preferred travel date to perform a flight search.

Normal

The customer searches for available flights through the airline booking system. The customer enters the departure city, destination city, travel date, and number of passengers in the search form. The system validates the entered search criteria and retrieves matching flights from the flight database. The system then displays a list of available flights including airline name, departure and arrival times, ticket classes, and prices. The customer reviews the search results and selects a preferred flight to view its detailed information. The search process is completed successfully, and the available flights are presented to the customer.

Alternative

The customer searches for available flights but follows an alternative search process. After viewing the initial search results, the customer refines the search by applying filters such as ticket class, price range, airline, or departure time. The system updates the search results dynamically based on the selected filters. The customer may also modify the search criteria, such as changing the travel date or destination, and perform the search again. The system retrieves and displays the updated list of available flights according to the new criteria, allowing the customer to continue browsing flight options.

Error

The customer searches for available flights by entering valid search criteria, but the system does not find any matching flights for the selected date or route. The system displays an error message indicating that no flights are available and suggests searching with alternative dates or destinations. The search process ends without displaying any available flight results.

Error

The customer attempts to search for available flights, but the system fails to retrieve flight information due to a database connection or system service failure. The system displays an error message informing the customer that the service is temporarily unavailable and advises them to try again later. No flight data is displayed, and the search request is not completed successfully.

2.3.5 Scenario #5: Cancel Flight Booking and Process Refund [Ayham]

Initial Assumption

The customer is a registered user of the airline online booking system and is successfully logged in. The customer has an existing confirmed booking with a valid reservation reference (PNR). The ticket is within the allowed cancellation period according to airline policies. The system is operational and connected to the airline reservation system and financial/payment services. The customer has access to the internet and valid booking information.

Normal

The customer cancels a flight booking through the airline online booking system. The customer accesses the "My Bookings" section and enters the reservation reference (PNR) to retrieve the booking details. The system displays the flight information, ticket class, payment status, and applicable cancellation policy. The customer selects the cancel booking option and confirms the cancellation request. The system verifies the cancellation eligibility, calculates the refundable amount according to airline rules, and processes the cancellation. The booking status is updated to cancelled, and the refundable amount is returned to the customer through the original payment method. The system sends a cancellation confirmation and refund details to the customer via the system and email. The cancellation is successfully stored in the system database.

Alternative

The customer cancels a flight booking, but the ticket is partially refundable according to the airline policy. After confirming the cancellation request, the system calculates a cancellation penalty or service fee and displays the reduced refundable amount. The customer accepts the refund conditions and confirms the cancellation. The system processes the cancellation, updates the booking status to cancelled, and issues a partial refund to the customer. A cancellation confirmation with refund details is sent to the customer, and the transaction is completed successfully.

Error

The customer attempts to cancel a flight booking, but the ticket is not eligible for cancellation according to airline policy (e.g., non-refundable ticket or cancellation deadline passed). The system displays an error message explaining the reason, the booking remains unchanged, and no refund is issued.

Error

The customer submits a valid cancellation request, but the refund process fails due to a payment gateway or system connectivity error. The system notifies the customer, advises retrying later or contacting support, and rolls back the transaction without updating the booking or issuing a refund.

2.4 Effort / Time Estimation Calculation (MODIFIED)

[Leader: Diaa, Reviewing: Nasri, Discussing: Sameer, Finalizing: Ayham, Re-drawing: Omar]

2.4.1 Effort Estimation per User Requirement

UR	**Estimated number of Developers**	**Estimated Effort**	**Total effort (pw)**
UR0	2	1 week	$2 \times 1 = 2$ pw
UR1	2	1 week	$2 \times 1 = 2$ pw
UR2	3	2 weeks	$3 \times 2 = 6$ pw
UR3	3	2 weeks	$3 \times 2 = 6$ pw
UR4	2	1 week	$2 \times 1 = 2$ pw
UR5	2	1 week	$2 \times 1 = 2$ pw
UR6	1	1 week	$1 \times 1 = 1$ pw
UR7	2	2 weeks	$2 \times 2 = 4$ pw
UR8	2	1 week	$2 \times 1 = 2$ pw
Total effort / avg	$(2+2+3+3+2+2+1+2+2)/9 = 2.1$ dev Estimated duration = 12 weeks		**27 pw**
Schedule time (30%)	Team Size = 5 developers → Parallel execution: $27 \div 5 = 5.4$ weeks Schedule buffer (30%) applied: $5.4 \times 1.3 \approx 7$ weeks (minimum) Worst-case sequential estimate: $12 \times 1.3 \approx 16$ weeks (maximum)		
Cost	Average salary = \$300 per developer per week Total cost = $300 \times 27 =$ \$8,100		
Profit Margin	Minimum (10%) → $8100 \times 1.1 =$ \$8,910 Maximum (30%) → $8100 \times 1.3 =$ \$10,530		

Table 1: Effort, Schedule, and Cost Estimation Summary

2.4.2 Task Assignment and Scheduling

To ensure timely and efficient implementation of the system, the project will be developed by a team of five developers working in parallel. The user requirements are distributed across the developers in a balanced manner, taking into account the estimated effort for each requirement. Tasks are assigned sequentially per developer while allowing maximum concurrency between team members.

User Requirement	Assigned Developers
UR0 (2 devs, 1 pw)	Dev1 (1 pw), Dev2 (1 pw)
UR1 (2 devs, 1 pw)	Dev3 (1 pw), Dev4 (1 pw)
UR2 (3 devs, 2 pw)	Dev1 (2 pw), Dev2 (2 pw), Dev3 (2 pw)
UR3 (3 devs, 2 pw)	Dev1 (2 pw), Dev4 (2 pw), Dev5 (2 pw)
UR4 (2 devs, 1 pw)	Dev2 (1 pw), Dev3 (1 pw)
UR5 (2 devs, 1 pw)	Dev4 (1 pw), Dev5 (1 pw)
UR6 (1 dev, 1 pw)	Dev5 (1 pw)
UR7 (2 devs, 2 pw)	Dev1 (2 pw), Dev2 (2 pw)
UR8 (2 devs, 1 pw)	Dev3 (1 pw), Dev4 (1 pw)

Table 2: Task Assignment and Scheduling

2.4.3 Developer-wise Task Allocation

Developer	Assigned User Requirements	Total Effort (pw)
Dev1	UR0 (1 pw), UR2 (2 pw), UR3 (2 pw), UR7 (2 pw)	7 pw
Dev2	UR0 (1 pw), UR2 (2 pw), UR4 (1 pw), UR7 (2 pw)	6 pw
Dev3	UR1 (1 pw), UR2 (2 pw), UR4 (1 pw), UR8 (1 pw)	5 pw
Dev4	UR1 (1 pw), UR3 (2 pw), UR5 (1 pw), UR8 (1 pw)	5 pw
Dev5	UR3 (2 pw), UR5 (1 pw), UR6 (1 pw)	4 pw

Table 3: Developer-wise Task Allocation

2.4.4 Discussion

Team Utilization The project utilizes all five team members to optimize development time. By distributing tasks in parallel across developers, the total development time is reduced from 12 weeks (sequential) to approximately 7 weeks (parallel with buffer), ensuring completion within the 14-week semester timeframe.

Workload Balance The workload is distributed as evenly as possible among developers:

- Dev1: 7 pw (handles complex requirements UR2, UR3, UR7)
- Dev2: 6 pw
- Dev3: 5 pw
- Dev4: 5 pw
- Dev5: 4 pw (available for testing and support)

Semester Feasibility

- Minimum time: 7 weeks (parallel development + 30% buffer)
- Maximum time: 16 weeks (sequential development + 30% buffer)
- Semester duration: 14 weeks

Conclusion: The project is feasible within semester constraints.

2.4.5 Cost Summary

Item	**Value**
Total Effort	27 person-weeks
Base Cost	$8,100
Minimum Project Cost (10% profit)	$8,910
Maximum Project Cost (30% profit)	$10,530

Table 4: Cost Summary

2.4.6 Risk Mitigation

- Dev5 has a lighter workload and can provide support if delays occur.
- A 30% schedule buffer accounts for unforeseen risks.
- Complex requirements (UR2, UR3) are assigned to multiple developers to ensure timely completion.

2.5 Actors Analysis and Their Description [MODIFIED]

[Leader: Nasri, Reviewing: Omar, Discussing: Diaa, Finalizing: Sameer, Re-drawing: Ayham]

Actor	**Semantics / Description**
Customer	Represents end users of the airline booking system. The Customer can search for available flights by specifying departure location, destination, travel date, and number of passengers; book flights by providing passenger information, selecting seat class and optional services, and completing payment via online card or pay-later option; modify bookings by changing flight dates, seat class, or optional services, including critical case modifications with proof document upload; cancel confirmed bookings and request refunds according to cancellation policies; complete pending payments at airline offices; view booking details using booking reference and passenger name; and submit support requests and track their status.
Staff	Represents airline employees responsible for supporting customers and managing booking operations. Staff can manage reservations, confirm offline (cash) payments at airline offices, issue and reissue electronic tickets, and complete pending payments.
Airline System	An external system that provides core airline data and enforces business rules. It supplies real-time flight schedules, seat availability by class (Economy, Business, VIP), pricing, and fare rules. It enforces cancellation and modification policies, manages seat inventory with locking mechanisms to prevent double-booking, validates seat availability, and supports booking state management.
Payment Gateway	An external financial system that securely processes online card payments. It receives encrypted payment requests with cardholder details using industry-standard encryption, validates card details, authorizes or rejects transactions based on funds availability, and returns payment confirmation or rejection status with transaction reference.
Notification System	An external system responsible for delivering automated notifications to customers and staff. It sends booking confirmations with e-tickets, modification confirmations, cancellation notices with refund details, pending booking reminders, support ticket acknowledgments, and status updates via email and SMS.

Table 5: Actor Analysis (Modified)

2.6 Use-Case Diagram [Modified]

[Leader: Nasri, Reviewing: Ayham, Discussing: Sameer, Finalizing: Diaa, Re-drawing: Omar]

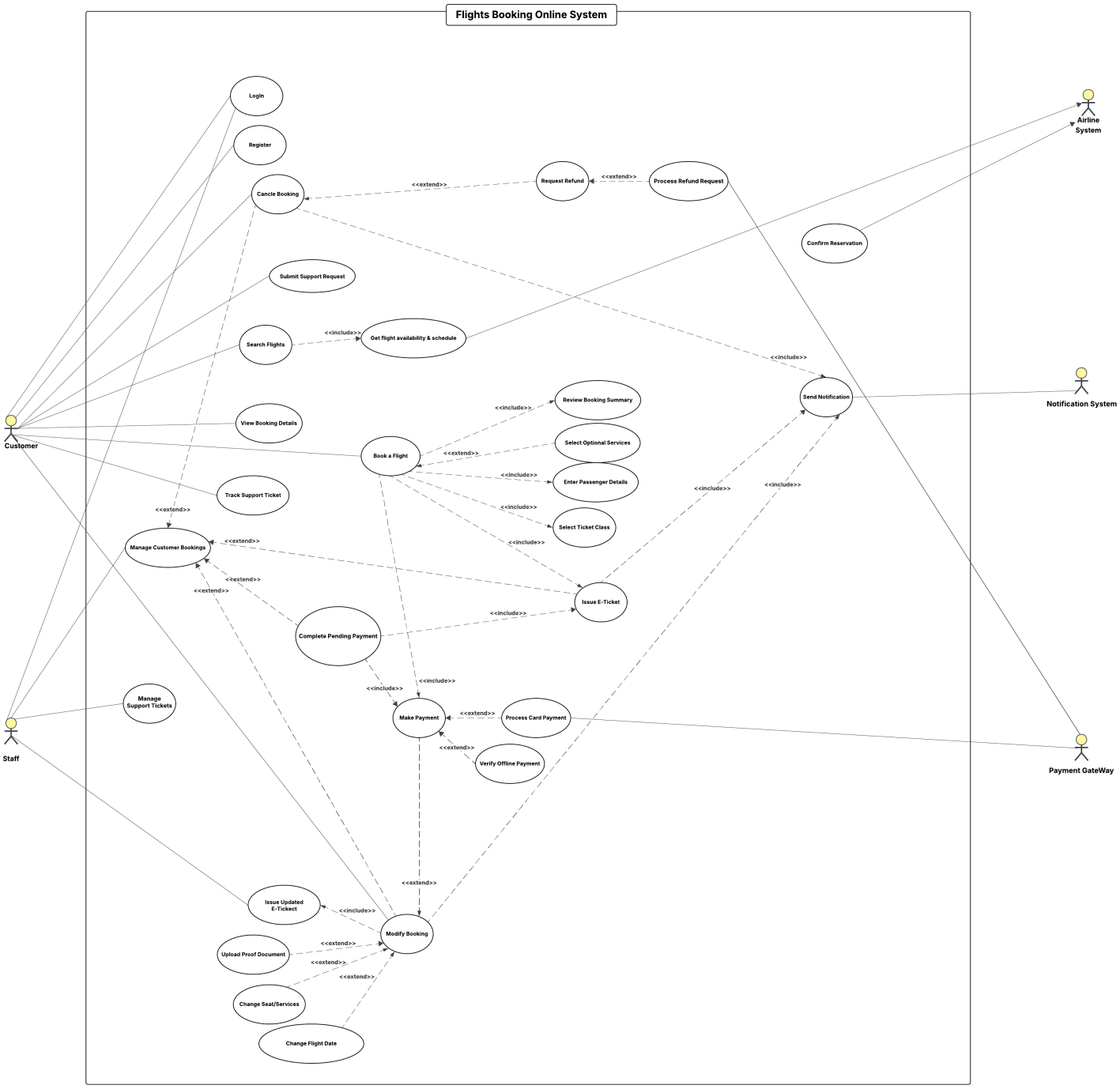


Figure 1: Use-Case Diagram

2.7 Description of Key Use-Cases

2.7.1 Use-Case #1: Book a Flight Ticket Online [Diaa]

Use Case Title	Book a Flight Ticket Online
Description	This use case describes how a registered Customer books a flight ticket online through the airline booking system. The Customer searches for available flights, selects a suitable flight and ticket class (VIP, Business, or Economy), enters passenger details, chooses optional services such as seat selection and baggage options, reviews the booking summary, and completes payment. If payment is successful, the system confirms the reservation, generates a booking reference, and issues an electronic ticket (e-ticket). The system also supports a pay-later option using cash payment through an airline office or authorized agent.
Actors	Primary Actor: Customer Secondary Actors: Payment Gateway, Notification System
Data	Flight search criteria (departure city, destination, travel date, number of passengers); Flight details (schedule, price, ticket class, seat availability); Passenger information (name, ID/passport, contact details); Optional services (seat selection, baggage options); Payment information (payment method, transaction status); Booking record (reservation reference, booking status, ticket status).
Stimulus / Trigger	The Customer selects the "Book / Reserve" option for a chosen flight after viewing available flight search results.
Pre-conditions	1. The Customer is registered and successfully logged in to the system. 2. The system is operational and connected to the flight database and payment services. 3. There are scheduled flights available for booking. 4. The Customer has valid personal and travel information required by airline policies.
Sequence / Flow of Events	**Normal Flow (Successful):** 1. The Customer enters flight search details (departure city, destination, travel date, number of passengers). 2. The system retrieves and displays available flights with schedules, prices, and ticket classes. 3. The Customer selects a flight and chooses a ticket class (Economy, Business, or VIP).

	4. The system prompts the Customer to enter passenger details and select optional services. 5. The Customer enters passenger information and selects optional services such as seat selection and baggage options. 6. The system validates the entered information and checks seat and service availability. 7. The system calculates the total price and displays the booking summary. 8. The Customer confirms the booking summary and selects **online card payment**. 9. The system sends the payment request to the Payment Gateway. 10. The Payment Gateway authorizes the transaction. 11. The system finalizes the booking, generates a reservation reference number, and issues the electronic ticket. 12. The system stores the booking with status **Confirmed** and ticket status **Issued**. 13. The system sends booking confirmation and e-ticket to the Customer via the Notification System. **Alternative Flow (Successful – Pay Later):** • **8A1.** At step 8, the Customer selects **pay later / cash payment**. • **9A1.** The system temporarily reserves the seats and creates a booking with status **Pending Payment**. • **10A1.** The system displays instructions for completing payment at an airline office or authorized agent. • **11A1.** After payment is confirmed by staff, the system updates the booking to **Confirmed** and issues the e-ticket. **Error Flow 1 (Payment Failure):** • **9E1.** The Payment Gateway rejects the transaction due to authorization failure or insufficient funds. • **10E1.** The system displays a payment failure message and does not finalize the booking. • **11E1.** The system releases reserved seats after a timeout and no ticket is issued. **Error Flow 2 (System Failure):** • **11E2.** The system fails to save the booking due to a database or connectivity error. • **12E2.** The system notifies the Customer and rolls back the transaction to prevent inconsistent data.

Post-conditions / Response	**Successful**: A booking record exists with a unique reservation reference. Booking status is Confirmed (or Pending Payment until paid in the alternative flow); ticket status is Issued once payment is confirmed; selected services are stored; confirmation and e-ticket are sent to the Customer. **Unsuccessful**: No confirmed booking exists; no e-ticket is issued; any temporary seat reservations are released; failure details are logged by the system.
Comments	Online payments must be processed through a secure payment gateway. Cash payments are confirmed by airline staff. Booking finalization must be atomic to avoid partial or inconsistent reservations.

2.7.2 Use-Case #2: Change Flight Date [Nasri]

Use Case Title	Change Flight Date
Description	This use case describes how a registered Customer changes the travel date of an already Confirmed (paid) booking due to a critical case (medical emergency). The Customer selects a booking, submits the critical-case reason and supporting document if required, chooses a replacement flight on the same route/class, pays any fees/fare difference, and receives an updated e-ticket/itinerary. The system updates booking and seat inventory atomically and notifies the Customer.
Actors	Primary Actor: Customer Secondary Actors: Payment Gateway, Notification System, Sales Staff
Data	Booking reference (PNR) and passenger last name, booking status, fare rules/change window, critical-case reason and proof document, available flights (schedule, class, seats), change fee and fare difference, payment transaction status, updated itinerary/e-ticket, audit/modification log, notification messages.
Stimulus / Trigger	The Customer selects a confirmed booking in "My Bookings" and clicks "Change Flight Date."
Pre-conditions	1. The Customer is registered and successfully logged in to the system. 2. A valid booking exists with status Confirmed (Paid) and has a valid booking reference (PNR). 3. The booking is within the allowed change window, and fare rules allow date change (critical cases permitted). 4. The system is operational and connected to the flight database/inventory subsystem and payment services. 5. No other modification request is currently in progress for the same booking (booking is not locked).
Sequence / Flow of Events	**Normal Flow (Successful - Card Payment):** 1. The Customer opens "My Bookings" and selects the booking to modify. 2. The system retrieves booking details and displays the change policy (window, fees, fare difference rules). 3. The Customer selects "Critical Case", enters the reason, and uploads proof (if required). 4. The system validates eligibility and locks the booking to prevent parallel modifications.

	5. The Customer selects the new travel date. 6. The system displays available replacement flights for the same route with schedules, classes (Economy/Business/VIP), and seat availability. 7. The Customer selects a replacement flight (same class by default). 8. The system re-checks availability and validates fare rules for the requested change. 9. The system calculates the change fee and fare difference and displays a summary (old/new itinerary, total amount due). 10. The Customer confirms and selects card payment. 11. The system sends the payment request to the Payment Gateway/Bank. 12. The Payment Gateway authorizes the transaction successfully. 13. The system finalizes the modification: releases old seat, reserves new seat, updates the booking record, stores a modification log, issues the updated e-ticket, and notifies the Customer (email/SMS). **Alternative Flow (Successful – Pay Later / Cash or Bank Transfer at Agency):** • **8A1.** After step 9 in the normal flow, the Customer selects Pay Later (Cash at Agency / Bank Transfer) instead of card payment. • **9A1.** The system places a temporary hold on the selected new seat and creates a change request with status **Pending Payment**. • **10A1.** The system displays instructions to pay at the airline office/authorized agent or upload bank transfer proof within the allowed time window. • **11A1.** Customer Support Staff verifies the payment and records confirmation in the system. • **12A1.** The system confirms the modification, updates booking and seat inventory, issues the updated e-ticket, and notifies the Customer. **Error Flow 1 (Payment Failure):** • **11E1.** The Payment Gateway/Bank rejects the card transaction (authorization failure, insufficient funds, or timeout). • **12E1.** The system displays a payment failure message and records the failed attempt; the modification is not confirmed. • **13E1.** The original booking remains unchanged; any temporary seat holds are released after timeout; the Customer may retry or choose another payment method or contact support.

	Error Flow 2 (System Failure – DB/Inventory Update Failure): • **13E2.** Payment is successful, but the system fails to save/update the booking or seat inventory due to a database/connectivity error or inventory service unavailability. • **14E2.** The system prevents inconsistent data by rolling back the update or marking the request as **Pending Verification**; no updated ticket is issued until resolved. • **15E2.** The system notifies the Customer with a support reference number and alerts support/finance to verify booking and payment status.
Post-conditions / Response	**Successful**: Booking itinerary is updated; new seat is reserved and old seat is released; modification log is stored; updated e-ticket/itinerary is issued and sent to the Customer. **Unsuccessful**: Booking remains unchanged (or marked Pending Verification in system failure cases); no updated e-ticket is issued; holds are released; failures are logged; Customer is notified with next actions.
Comments	Booking modification should be atomic to avoid partial updates. The system should lock the booking during the change process. Payment status must be verified to avoid duplicate charges. Critical-case documents must be stored securely and accessed only by authorized staff.

2.7.3 Use-Case #3: Complete Pending Booking (Pay Later / Cash Payment at Office) [Omar]

Use Case Title	Complete Pending Booking (Cash Payment Confirmation)
Description	This use case describes how a registered Customer completes an already-created booking that is in Pending Payment status by paying at the airline office/authorized agent. The Customer provides the reservation reference (PNR) and identification, the Staff retrieves and verifies the booking, collects cash payment (or verifies bank transfer proof), confirms payment in the system, and the system updates the booking to Confirmed, issues an electronic ticket (e-ticket), generates the invoice/receipt, stores the transaction, and notifies the Customer.
Actors	Primary Actor: Sales Staff (Office/Agent) Secondary Actors: Customer, Notification System, (optional) Finance/Manager
Data	Reservation reference (PNR); Booking record (status, deadline, flight details, ticket class); Passenger identification details; Amount due; Payment method (cash / bank transfer); Payment confirmation record; Booking status update (Pending Payment → Confirmed); E-ticket/itinerary; Invoice/receipt; System logs; Notification message details.
Stimulus / Trigger	The Customer visits the airline office/authorized agent and requests to complete payment for a Pending Payment booking by providing the PNR.
Pre-conditions	1. The Customer is registered and has a valid booking in the system with status Pending Payment. 2. The booking is within the allowed payment time window (not expired). 3. Sales Staff is authenticated and authorized to confirm payments and update booking status. 4. The system is operational and connected to the booking database and notification service.

Sequence / Flow of Events	**Normal Flow (Successful – Cash Payment):** 1. The Sales Staff opens the "Pending Bookings / Retrieve Booking" function. 2. The Customer provides the reservation reference (PNR) and identification. 3. The system retrieves the booking and displays booking details, total amount due, and payment deadline. 4. The Sales Staff verifies customer identity and checks booking validity (status Pending Payment, not expired). 5. The Customer pays the required amount in cash. 6. The Sales Staff records the cash payment and confirms it in the system. 7. The system updates booking status to **Confirmed** and stores the payment transaction. 8. The system issues the electronic ticket (e-ticket) and generates invoice/receipt. 9. The system sends booking confirmation and e-ticket to the Customer via the Notification System (email/SMS). 10. The system logs the completion for audit purposes. **Alternative Flow (Successful – Bank Transfer Proof):** • **5A1.** At step 5, the Customer pays via bank transfer and provides proof/receipt. • **6A1.** The Sales Staff verifies the transfer details (amount, reference, date) and confirms it (with Finance/Manager if required). • **7A1.** The system records the payment and updates booking status to **Confirmed**. • **8A1.** The system issues the e-ticket and sends confirmation notifications. **Error Flow 1 (Booking Expired):** • **3E1.** At step 3, the system finds the booking status is Expired/Cancelled (deadline passed). • **4E1.** The system displays an error message and prevents payment confirmation. • **5E1.** No ticket is issued; the Customer is instructed to create a new booking. **Error Flow 2 (System/Database Failure):** • **6E2.** At step 6, the system fails to save/update the booking due to database/connectivity error.

	• **7E2.** The system shows failure message and does not issue an e-ticket. • **8E2.** The system logs the error; booking remains **Pending Payment** until resolved (or verified and completed by staff).
Post-conditions / Response	**Successful**: Booking status becomes Confirmed; payment transaction is stored; e-ticket is issued; receipt is generated; notification is sent to Customer; audit log is updated. **Unsuccessful**: Booking remains Pending Payment or is Expired/Cancelled; no e-ticket is issued; any failure is logged and Customer is informed of next steps.
Comments	Payment confirmation must be performed only by authorized staff. The system must prevent issuing tickets before successful status update to avoid inconsistent reservations. Booking completion should be atomic and fully logged for auditing and dispute handling.

2.7.4 Use-Case #4: Search for Available Flights [Sameer]

Use Case Title	Search for Available Flights
Description	This use case describes how a registered Customer searches for available flights through the airline online booking system. The Customer enters flight search criteria (departure city, destination, travel date, and number of passengers). The system validates the input, retrieves matching flights from the flight database, and displays available options including schedules, prices, and ticket classes. The system also supports refining results using filters (e.g., class, price range, airline, departure time).
Actors	Primary Actor: Customer Secondary Actors: Airline online booking gateway, Notification gateway
Data	Flight search criteria (departure city, destination city, travel date, number of passengers); Filter/sort criteria (ticket class, price range, airline, departure time); Flight results data (flight number, airline, schedule, availability, price, ticket classes).
Stimulus / Trigger	The Customer selects the "Search Flights" option and submits the search form.
Pre-conditions	1. The Customer is registered and can access the system (logged in). 2. The system is operational and connected to the airline booking gateway. 3. Flight schedules, availability, and prices are stored and up-to-date in the airline booking gateway. 4. The Customer has valid travel search information (route, date, passengers).
Sequence / Flow of Events	**Normal Flow (Successful):** 1. The Customer opens the flight search page. 2. The Customer enters flight search details (departure city, destination, travel date, number of passengers). 3. The Customer clicks Search. 4. The system validates the search input (required fields and date validity). 5. The system queries the Flight Database for matching flights. 6. The Flight booking gateway returns available flights with schedules, prices, and ticket classes.

	7. The system displays the search results list to the Customer. 8. The Customer reviews the results and may open a selected flight to view details. **Alternative Flow (Successful – Refine Results):** • **7A1.** At step 7, the Customer applies filters (ticket class, price range, airline, departure time). • **8A1.** The system re-filters/re-sorts the results and displays the updated list. **Alternative Flow (Successful – Modify Search Criteria):** • **2A2.** At step 2, the Customer changes one or more search fields (e.g., date or destination). • **3A2.** The Customer submits the updated search request. • **4A2.** The system repeats steps 4–7 and displays new results. **Error Flow 1 (No Results Found):** • **5E1.** At step 5, the Flight booking gateway returns no matching flights for the entered criteria. • **6E1.** The notification gateway displays a message "No flights found" and suggests trying different dates or destinations. **Error Flow 2 (Booking Gateway Failure):** • **5E2.** At step 5, the system fails to retrieve flight results due to a flight booking gateway connectivity issue. • **6E2.** The notification gateway displays an error message indicating the service is temporarily unavailable and logs the failure.
Post-conditions / Response	**Successful**: A flight search result list is displayed containing available flights matching the criteria (and refined filters if applied). No booking is created, and the system is ready for the Customer to proceed to flight selection/booking. **Unsuccessful**: No flight results are displayed (either no matches or system failure). The system does not create any booking, and the Customer is prompted to retry or modify criteria. Any failure details are logged by the system.
Comments	Search input must be validated (e.g., travel date cannot be in the past). Flight results should be retrieved efficiently to ensure good performance under high search volume. Results must reflect real-time seat availability and updated pricing to avoid inconsistencies during later booking steps.

2.7.5 Use-Case #5: Cancel Flight Booking and Process Refund [Ayham]

Use Case Title	Cancel Flight Booking and Process Refund
Description	This use case describes how a registered Customer cancels an existing flight booking through the airline online booking system. The Customer retrieves the booking using a reservation reference (PNR), reviews the applicable cancellation policy, and submits a cancellation request. If eligible, the system processes the cancellation, calculates the refundable amount, updates the booking status, and issues a refund through the original payment method. A cancellation confirmation is then sent to the Customer.
Actors	Primary Actor: Customer Secondary Actors: Payment Gateway, Airline System, Notification System
Data	Booking reference (PNR); Flight details; Ticket class; Payment information; Cancellation policy rules; Refund amount; Booking status; Transaction status; Notification details.
Stimulus / Trigger	The Customer selects the "Cancel Booking" option for an existing confirmed reservation.
Pre-conditions	1. The Customer is registered and successfully logged in to the system. 2. The Customer has a valid confirmed booking with a reservation reference (PNR). 3. The system is operational and connected to airline reservation and payment services. 4. The booking is within the allowed cancellation period according to airline policy.
Sequence / Flow of Events	**Normal Flow (Successful Cancellation):** 1. The Customer accesses the My Bookings section. 2. The Customer enters the reservation reference (PNR). 3. The system retrieves and displays booking details and cancellation policy. 4. The Customer selects Cancel Booking. 5. The system validates cancellation eligibility. 6. The system calculates the refundable amount.

	7. The Customer confirms the cancellation request. 8. The system updates the booking status to **Cancelled**. 9. The system processes the refund through the original payment method. 10. The system records the cancellation and refund transaction. 11. The system sends cancellation confirmation and refund details to the Customer. **Alternative Flow (Partial Refund):** • **7A1.** The system determines that a cancellation penalty applies. • **8A1.** The system displays the reduced refundable amount. • **9A1.** The Customer accepts the refund conditions. • **10A1.** The system processes the cancellation and issues a partial refund. • **11A1.** The system sends updated cancellation confirmation. **Error Flow 1 (Cancellation Not Allowed):** • **5E1.** The system determines the ticket is non-refundable or outside the cancellation period. • **6E1.** The system displays an error message explaining the rejection. • **7E1.** The booking remains unchanged and no refund is issued. **Error Flow 2 (Refund Processing Failure):** • **9E2.** The payment gateway fails to process the refund. • **10E2.** The system notifies the Customer of the failure. • **11E2.** The system rolls back the transaction and logs the error.
Post-conditions / Response	**Successful**: Booking status is Cancelled; refund (full or partial) is issued; confirmation is sent to the Customer. **Unsuccessful**: Booking remains unchanged; no refund is issued; error details are logged and the Customer is notified.
Comments	Refund processing must comply with airline cancellation policies. Cancellation and refund operations must be atomic to avoid inconsistent booking or payment records.

2.8 Overall Activity Diagram [MODIFIED]

[Leader: Diaa, Reviewing: Ayham, Discussing: Nasri, Finalizing: Omar, Re-drawing: Sameer]

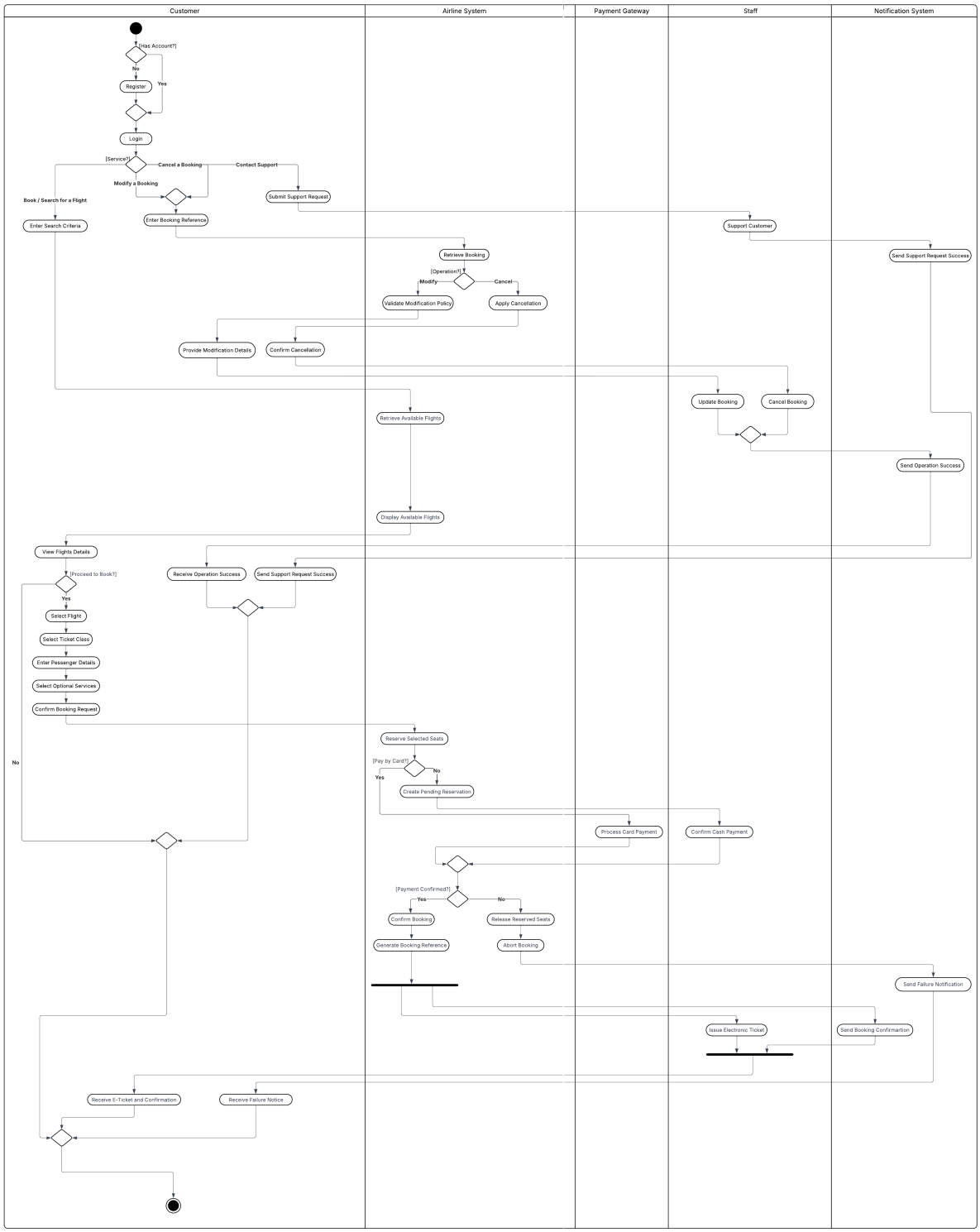


Figure 2: Overall Activity Diagram

2.9 Instance Activity Diagrams

2.9.1 Use-Case #1: Book a Flight Ticket Online [Diaa]

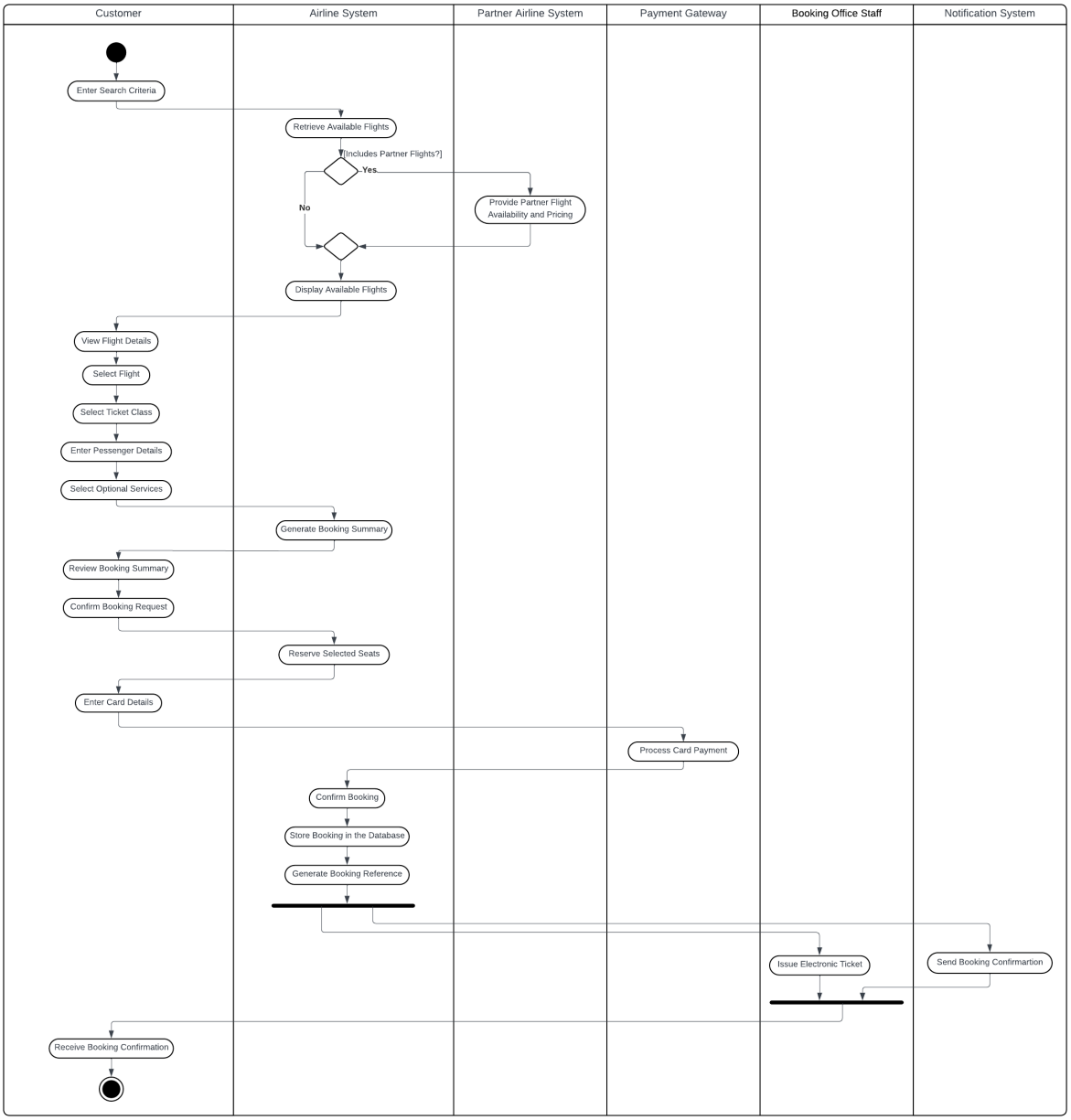


Figure 3: Activity Diagram – Book a Flight Ticket Online

2.9.2 Use-Case #2: Change Flight Date [Nasri]

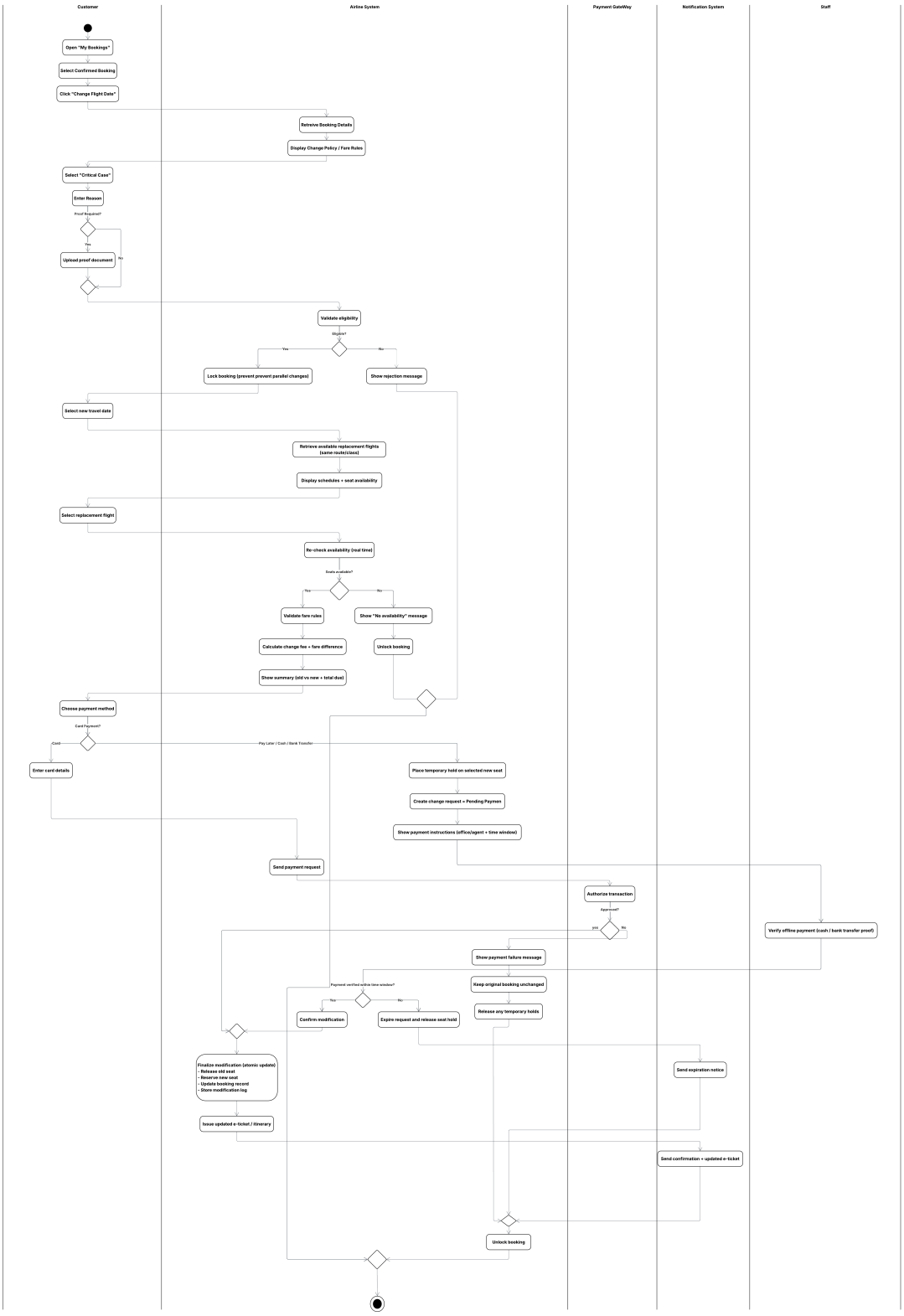


Figure 4: Activity Diagram – Change Flight Date

2.9.3 Use-Case #3: Complete Pending Booking (Pay Later / Cash Payment at Office) [Omar]

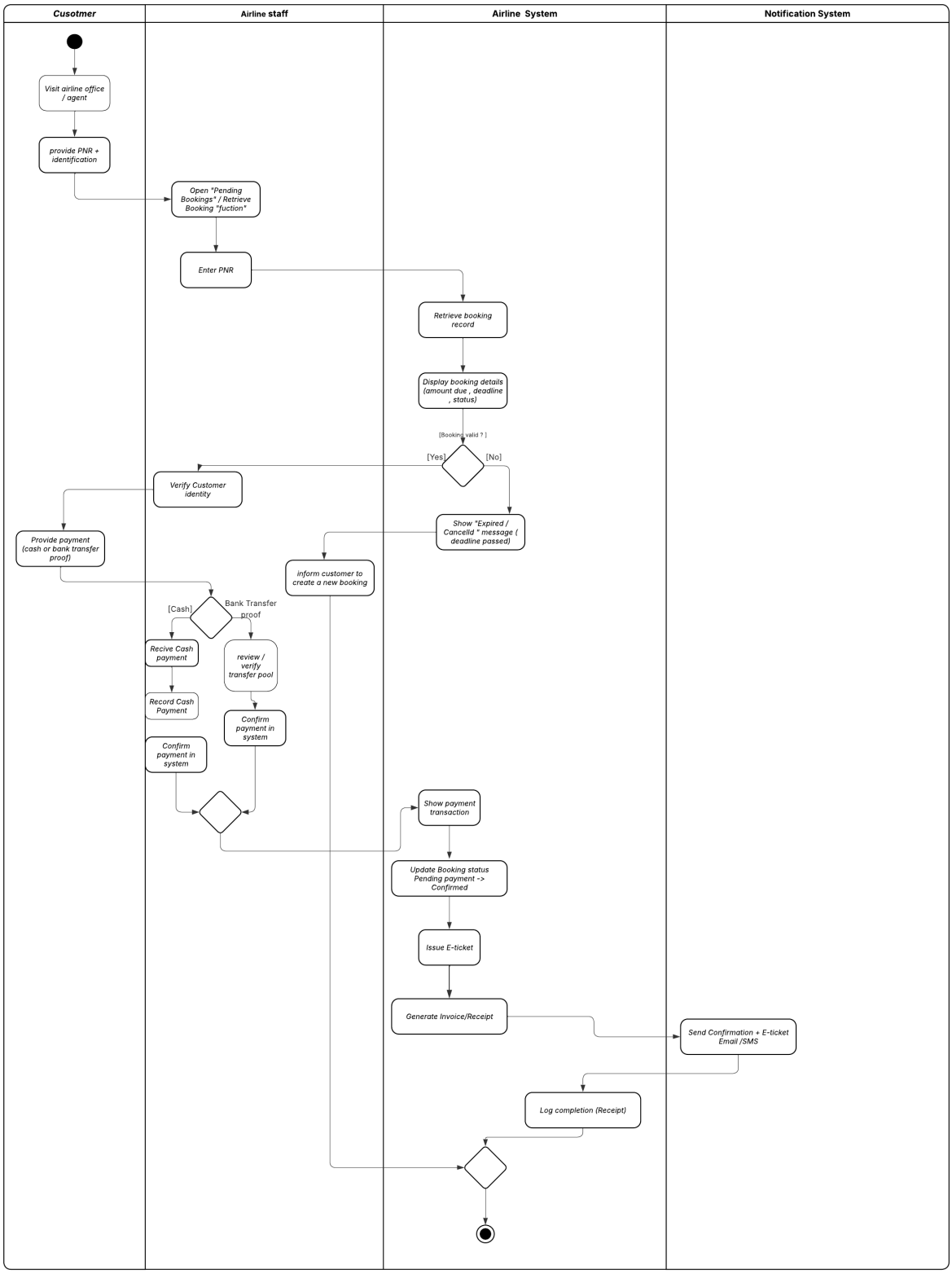


Figure 5: Activity Diagram – Complete Pending Booking

2.9.4 Use-Case #4: Search for Available Flights [Sameer]

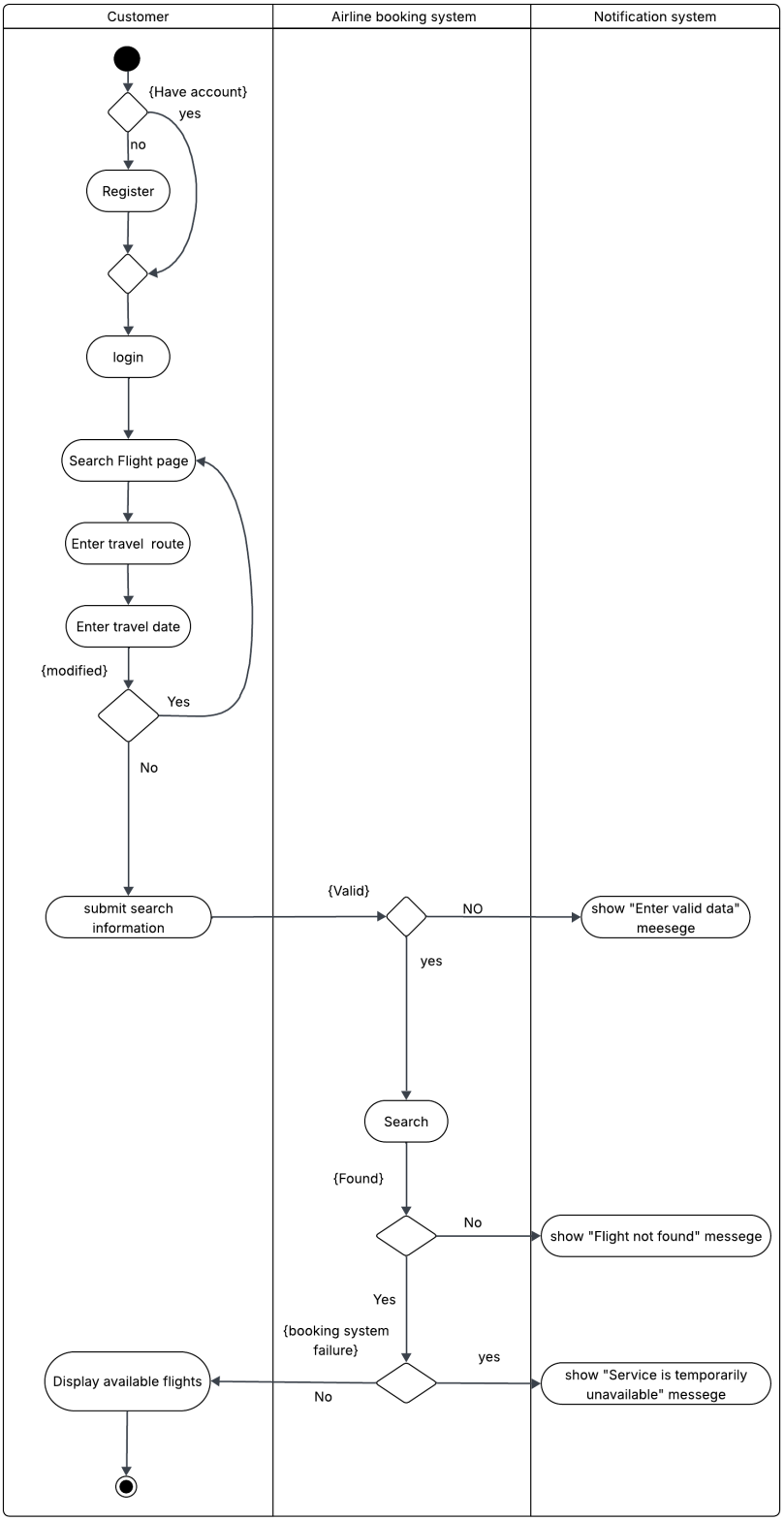


Figure 6: Activity Diagram – Search for Available Flights

2.9.5 Use-Case #5: Cancel Flight Booking and Process Refund [Ayham]

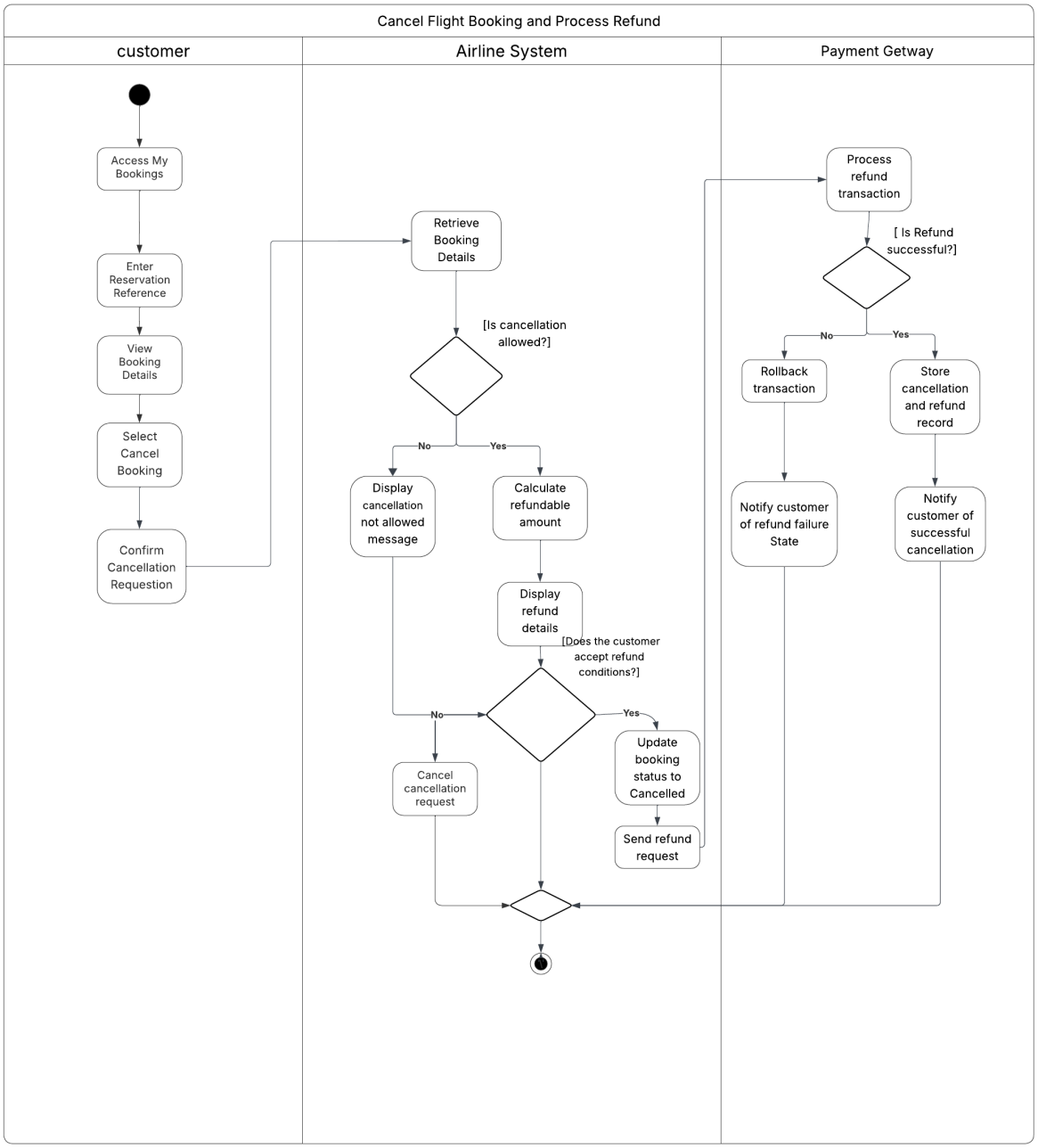


Figure 7: Activity Diagram – Cancel Flight Booking and Process Refund

3 System Analysis and Modelling

3.1 System Class Diagrams

3.1.1 Analysis Class Model

[Leader: Nasri, Reviewing: Diaa, Discussing: Omar, Finalizing: Sameer, Re-drawing: Ayham]

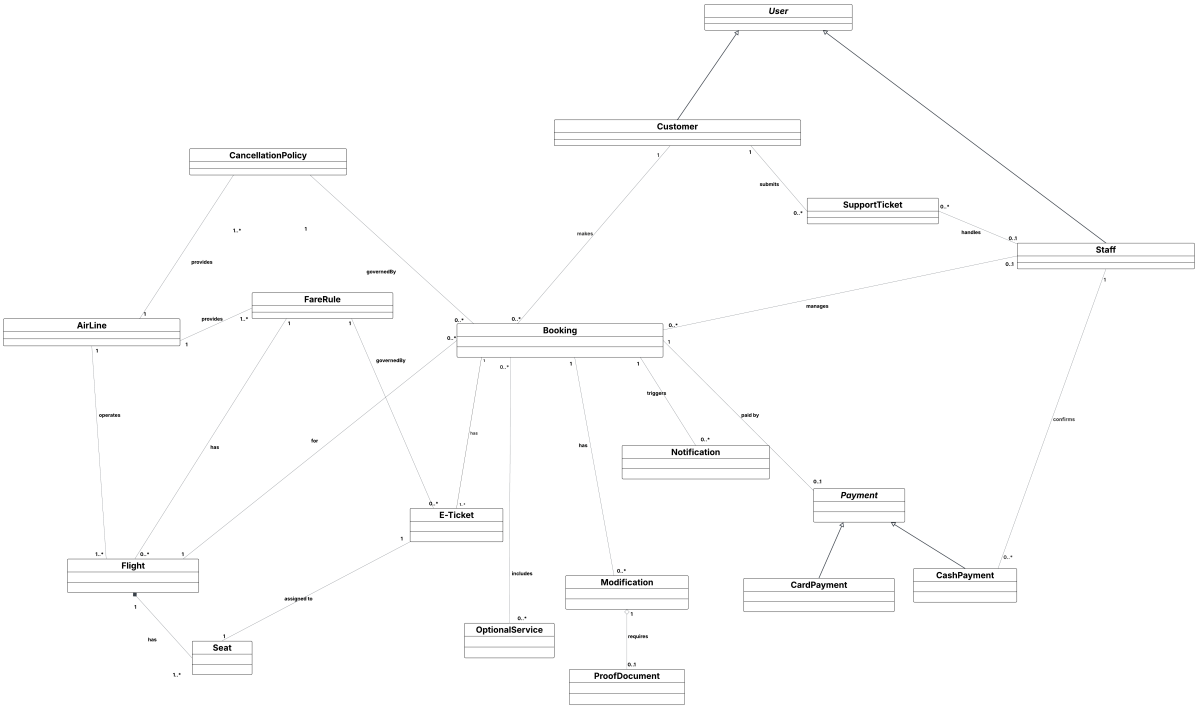


Figure 8: Analysis Class Model

3.1.2 Detailed Class Model

[Leader: Diaa, Reviewing: Nasri, Discussing: Omar, Finalizing: Sameer, Re-drawing: Ayham]

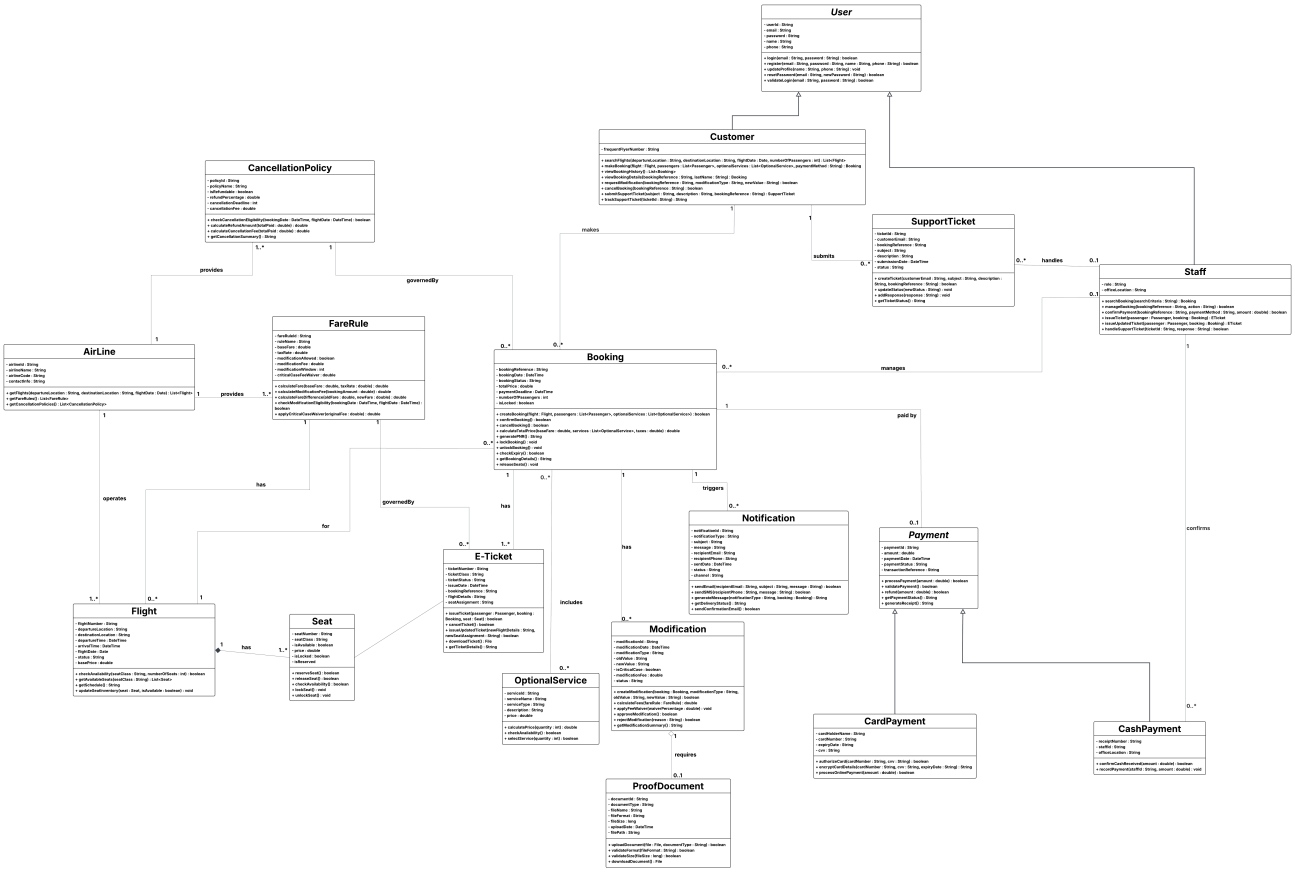


Figure 9: Detailed Class Model

3.2 Sequence Diagram

3.2.1 Use-Case #1: Book a Flight Ticket Online [Diaa]

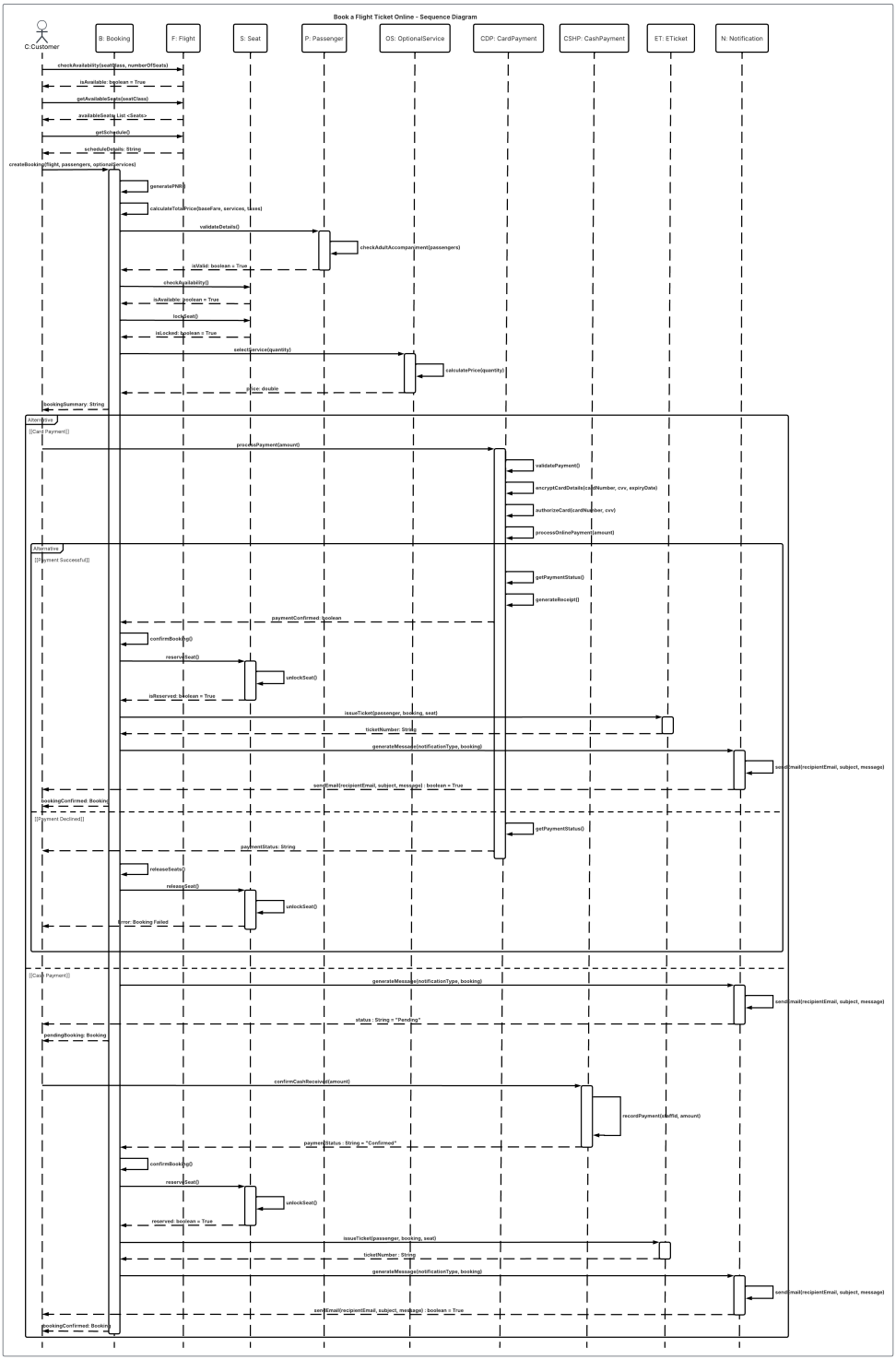


Figure 10: Sequence Diagram [Diaa]

3.2.2 Use-Case #2: Change Flight Date [Nasri]

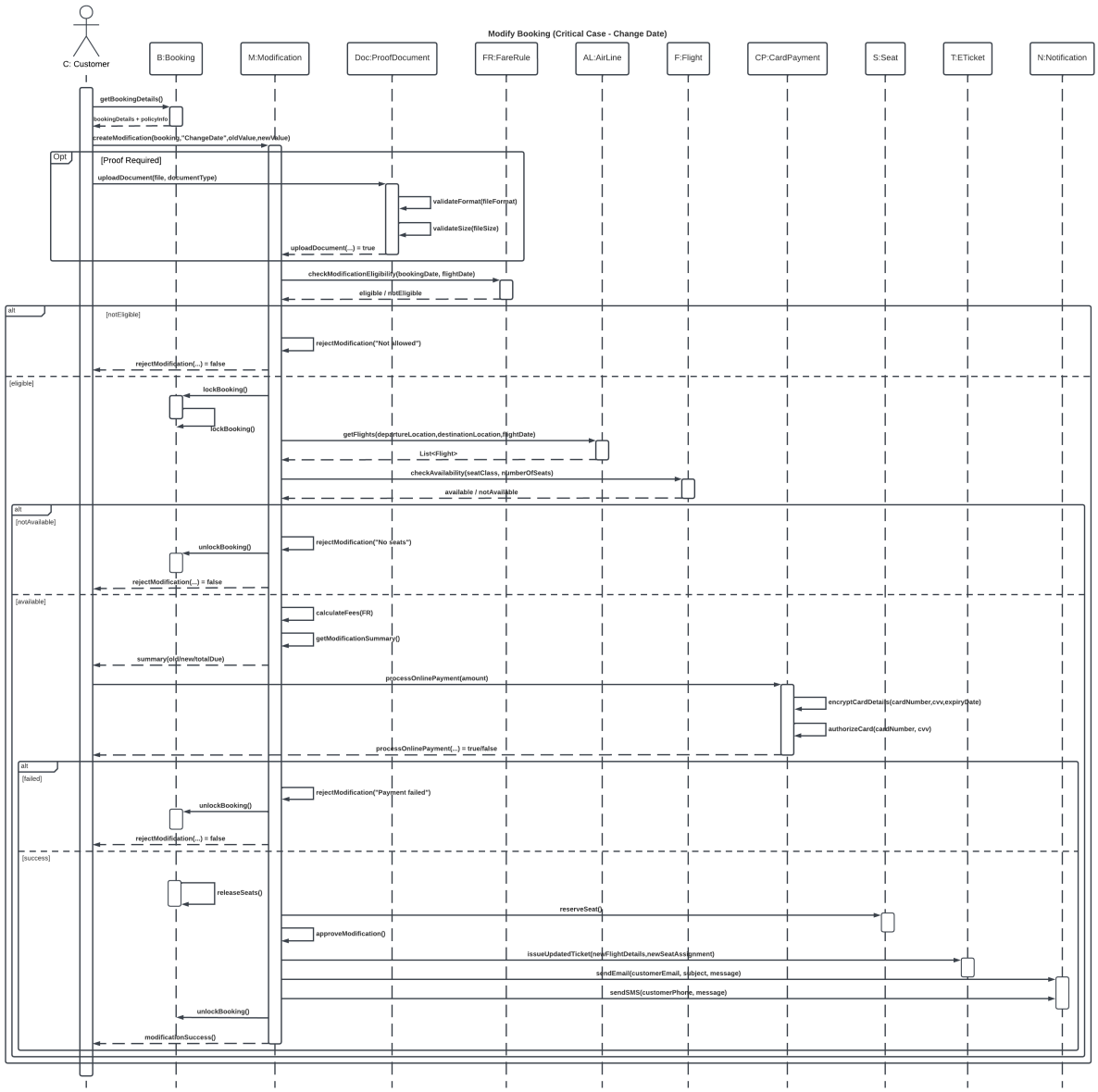


Figure 11: Sequence Diagram [Nasri]

3.2.3 Use-Case #3: Complete Pending Booking [Omar]

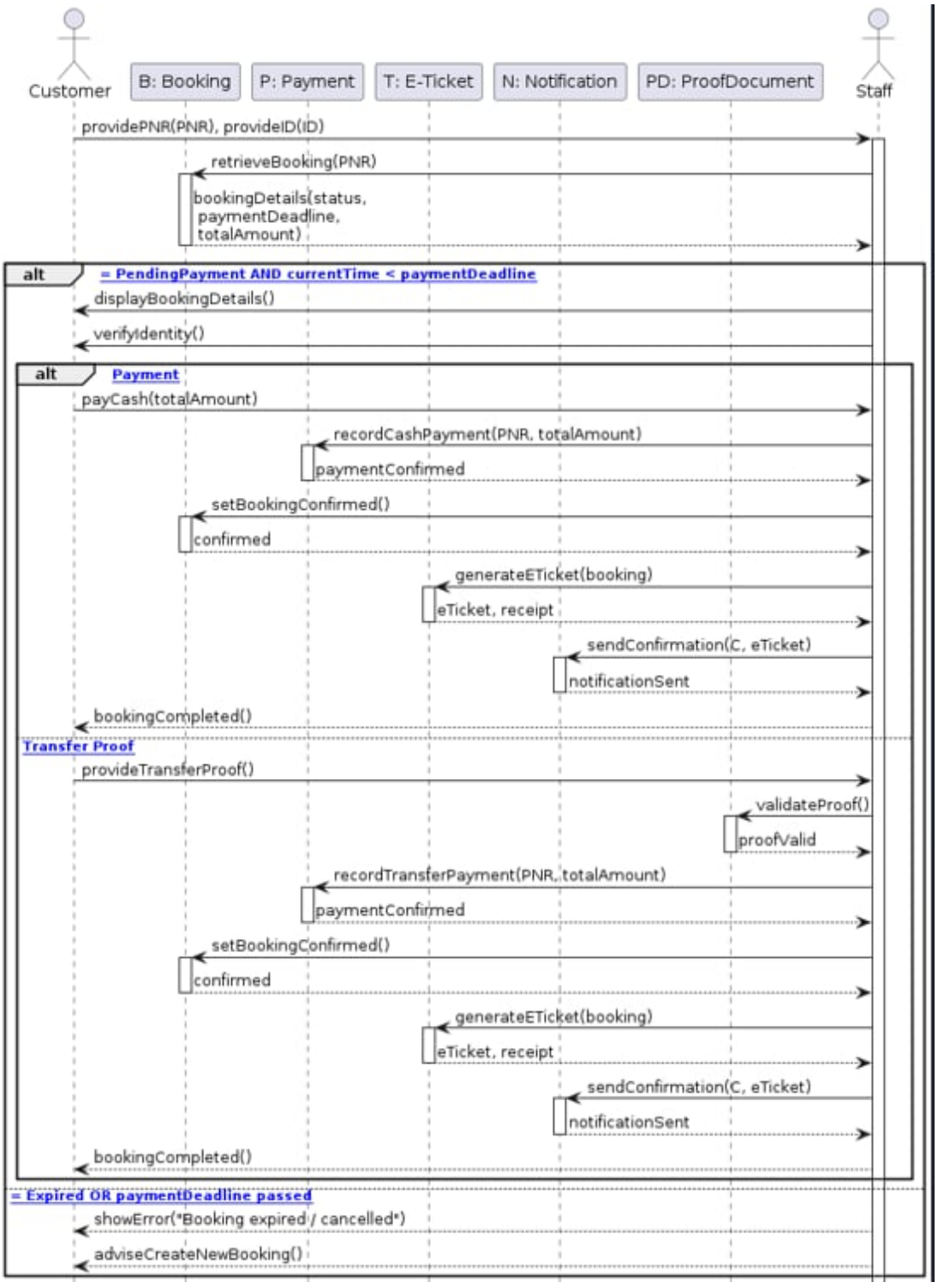


Figure 12: Sequence Diagram [Omar]

3.2.4 Use-Case #4: Search for Available Flights [Sameer]

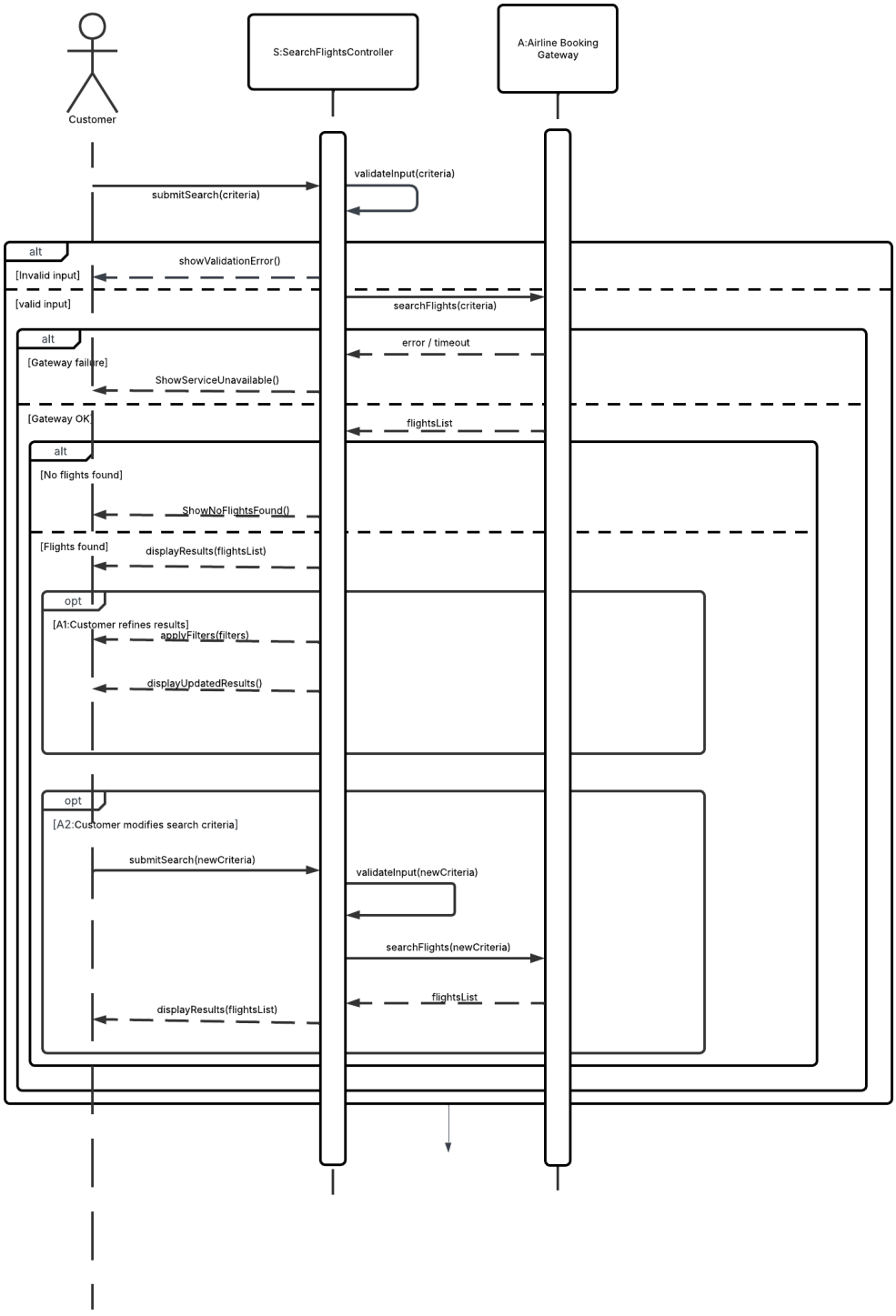


Figure 13: Sequence Diagram [Sameer]

3.2.5 Use-Case #5: Cancel Flight Booking and Process Refund [Ayham]

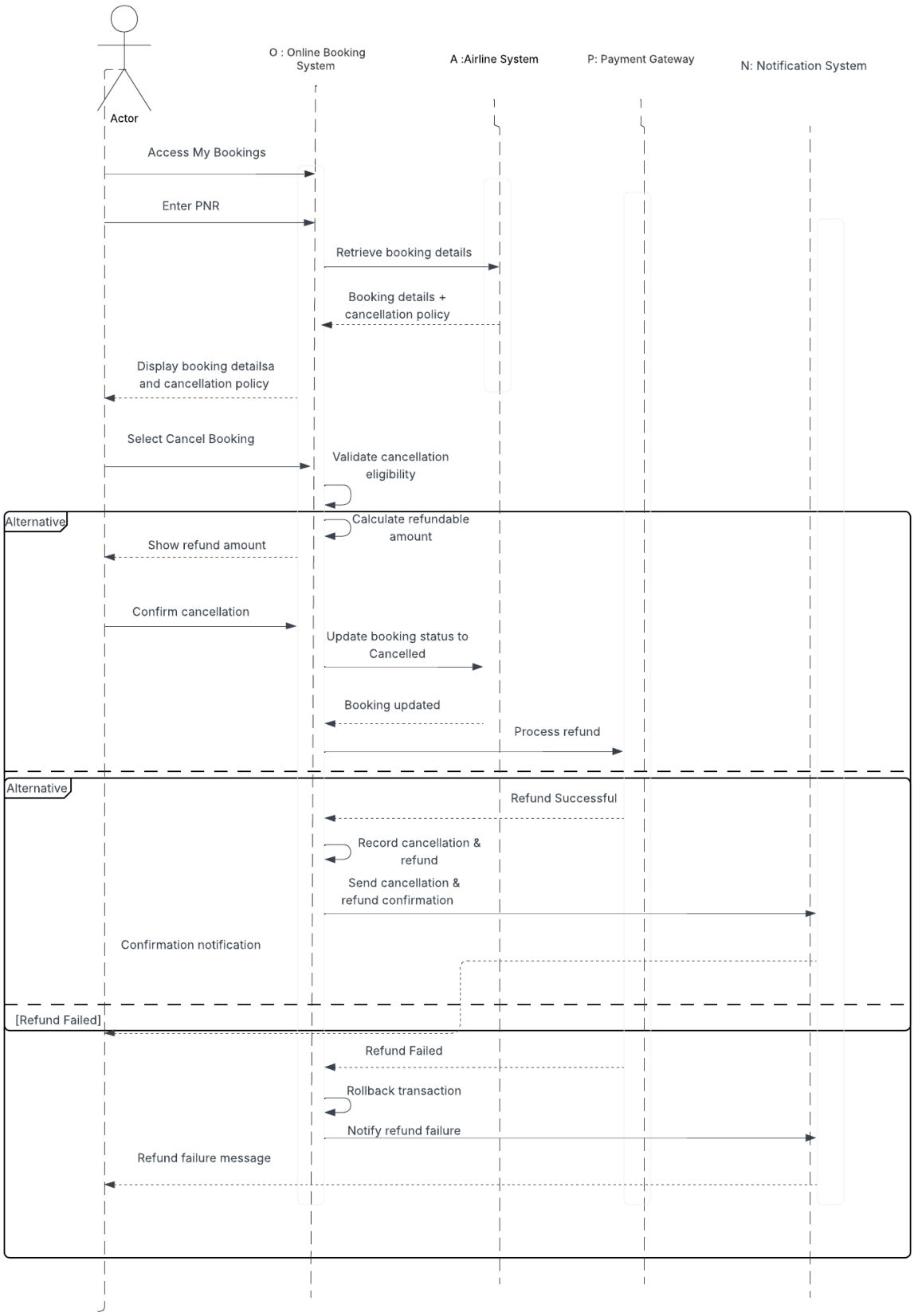


Figure 14: Sequence Diagram [Ayham]

4 System Design and Modelling

4.1 Description of Chosen Design Goals

[Leader: Diaa, Reviewing: Omar, Discussing: Nasri, Finalizing: Ayham, Re-drawing: Sameer]

Goal 1: High Cohesion

High cohesion means that each class has a single, well-defined responsibility where all its attributes and methods are closely related. In our class model, each class handles only one specific responsibility: Booking manages reservation lifecycle with attributes (bookingReference, bookingStatus, isLocked) and methods (createBooking(), cancelBooking(), lockBooking()); CancellationPolicy handles refund rules with calculateRefundAmount() and checkCancellationEligibility(); FareRule handles pricing with calculateFare() and calculateModificationFee(); and ProofDocument handles document validation with uploadDocument() and validateFormat(). At the component level, we organized related classes into cohesive components: Booking Component contains Booking, Passenger, and Modification; Payment Component contains Payment, CardPayment, and CashPayment; and Policy Component contains FareRule and CancellationPolicy.

Goal 2: Low Coupling

Low coupling means that classes have minimal dependencies and communicate only when necessary. In our class model, we used inheritance where Customer and Staff extend abstract User, and CardPayment and CashPayment extend abstract Payment, allowing components to work with abstract types. We used associations instead of embedding - Booking references CancellationPolicy through governedBy association rather than containing cancellation logic, so the Policy Component can be modified independently. We used aggregation where AirLine aggregates FareRule and CancellationPolicy, allowing reuse without tight coupling. At the component level, components communicate through well-defined interfaces: Booking Component calls Payment Component only through abstract Payment methods (processPayment(), refund()), and Notification Component is independent, receiving data through the triggers association.

Goal 3: Reliability (Fault Tolerance)

The system shall continue operating correctly and avoid breaking or going down when failures occur, while keeping data consistent (SR2.16). Reliability is achieved by ensuring all booking actions (create/modify/cancel) are atomic using the Persistence component's commit/rollback, so no partial bookings are saved. Booking states are persisted for recovery after crashes, and concurrency control (locking) prevents double-booking and conflicting updates.

4.2 Component Diagram

[Leader: Nasri, Reviewing: Ayham, Discussing: Diaa, Finalizing: Sameer, Re-drawing: Omar]

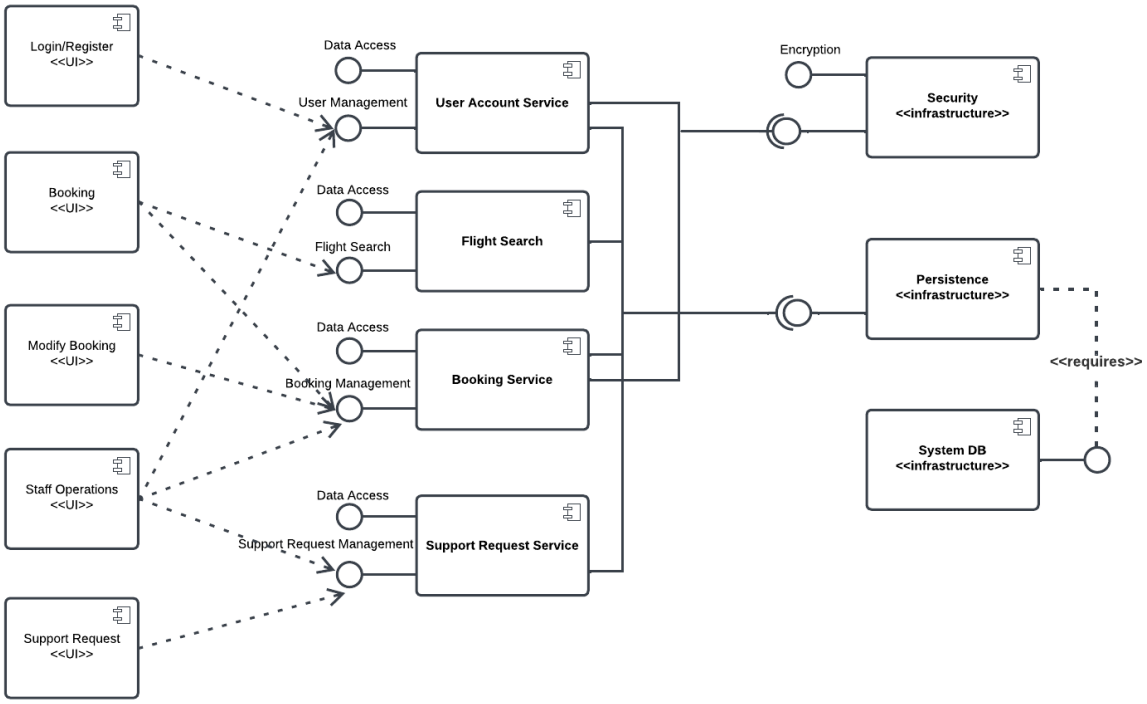


Figure 15: Component Diagram

4.3 Overall Architecture Diagram

[Leader: Sameer, Reviewing: Nasri, Discussing: Diaa, Finalizing: Ayham, Re-drawing: Omar]

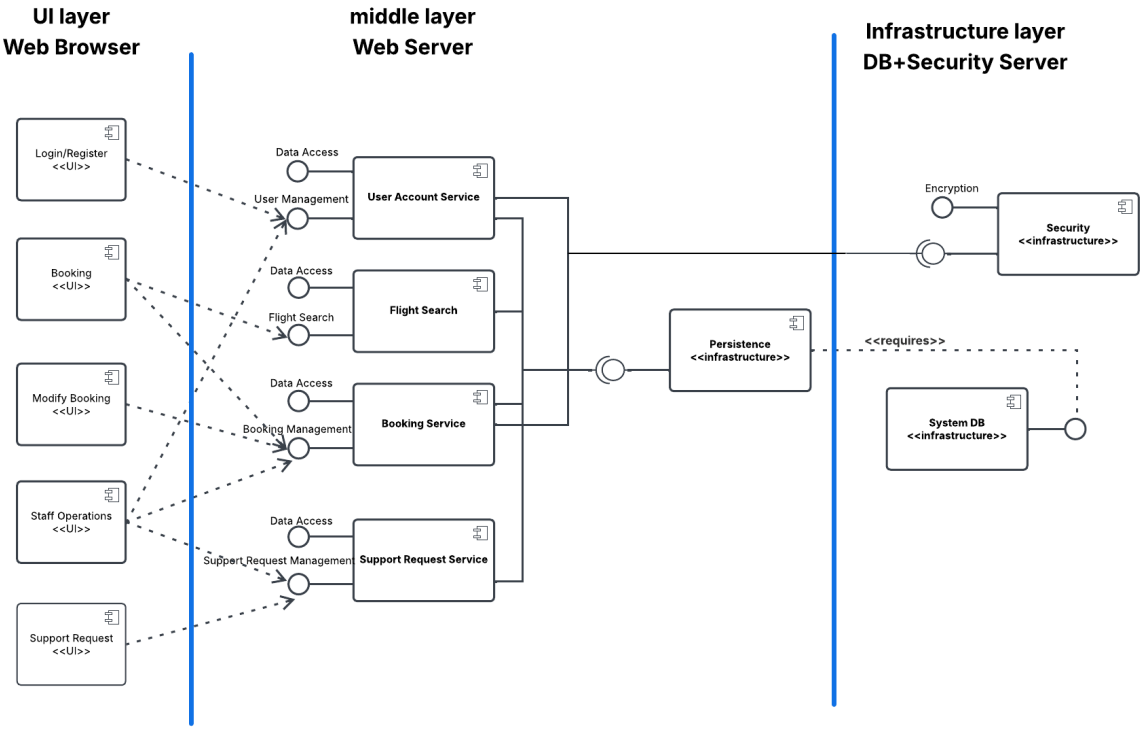


Figure 16: Overall Architecture Diagram

We selected a layered (3-tier client–server) architecture because it is the most suitable structure for an airline booking system where the user interface, business logic, and technical infrastructure must remain separated and easy to maintain. This architecture directly supports our design goals:

Low Coupling: The UI layer interacts only with the middle service layer through well-defined interfaces, and services access data only through the Persistence layer. This prevents direct dependencies between UI and database/security, so changes in one layer do not break the others.

High Cohesion: Each component in the service layer represents a single business service (User Account, Flight Search, Booking, Support Request), keeping responsibilities focused and avoiding mixing unrelated functionalities inside the same component.

Reliability (Fault Tolerance): Reliability is achieved by delegating data integrity to the infrastructure layer. The Booking Service performs create/modify/cancel operations through Persistence using atomic commit/rollback, and booking status and locks are stored in the database to prevent partial bookings, inconsistent states, and double-booking even if failures occur.

4.4 Deployment Diagram

[Leader: Nasri, Reviewing: Diaa, Discussing: Omar, Finalizing: Sameer, Re-drawing: Ayham]

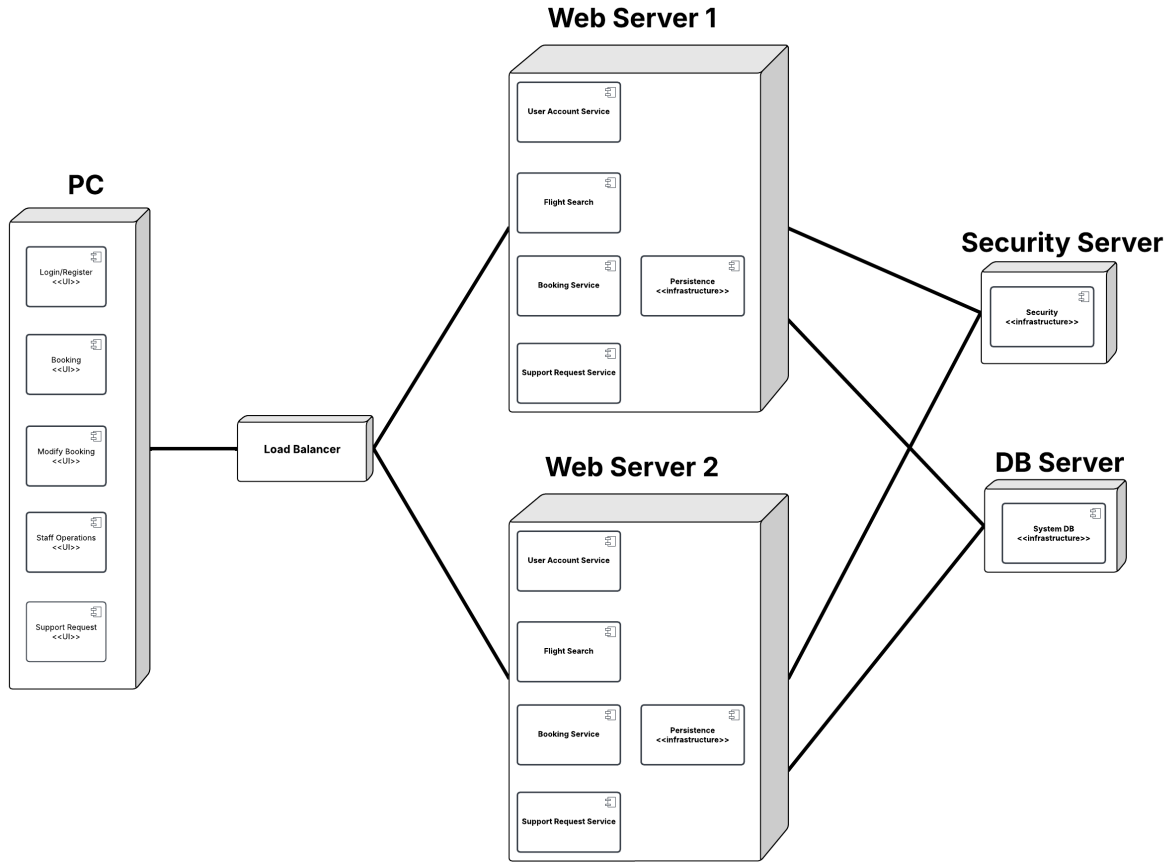


Figure 17: Deployment Diagram

Goal 3: Reliability (Fault Tolerance): Our deployment improves reliability by placing two web servers behind a load balancer, so if one web server fails the system can keep running using the other, reducing downtime, while booking data remains consistent by saving all booking actions through the Persistence layer to the DB (atomic commit/rollback and stored booking states). This setup also supports Goal 1 (High Cohesion) by keeping each node focused on a clear responsibility (UI on PC, services on web servers, security on security server, data on DB server), and Goal 2 (Low Coupling) by ensuring the PC connects only to the load balancer and the web servers interact with security and database through defined interfaces, making components easier to change and scale without breaking others.

A Appendix I

A.1 Minutes of Meetings with Customers

A formal meeting was held between the development team and the customer representatives to clarify expectations and gather detailed requirements for the Online Flight Booking System.

The meeting lasted 2 hour and covered essential business services, system functions, and operational processes required to ensure the system aligns with real airline reservation workflows.

Meeting Participants

From the Development Team:

- Diaa Badaha
- Nasri Omar
- Omar Shujaiah
- Sameer Ayman
- Ayham Amryah

From the Customer Side:

- Tareq Mansur
- Ali Hassouneh
- Alaa Awashra
- Noora Na'amneh
- Maysan Safi

Discussion Topics

1. Flight Search & Schedules:

The customers emphasized the need for users to easily search for flights using origin, destination, travel date, and passenger count. The system must show accurate schedules, prices, and seat availability in real time. Sorting and filtering (time, price, class) were suggested as future enhancements.

2. Booking Process & Passenger Information:

Customers clarified that the booking workflow must collect full passenger details, verify seat availability, and generate a reservation reference before payment. They also requested real-time seat availability checking to avoid double-booking issues.

3. Payment Methods & E-Ticket Issuance:

Multiple payment options are required, including:

- Credit/debit cards
- Cash at agency (offline)

The system must confirm successful payment and automatically issue an e-ticket via email. A clear payment confirmation page is also required.

4. Modifications & Cancellations:

Customers requested support for:

- Date changes

- Seat class upgrades/downgrades
- Optional services changes (e.g. baggage)

The system must enforce airline rules and calculate fees for each modification. For cancellations, the system must show refund eligibility and apply airline policy rules before processing.

5. Optional Travel Services:

Customers highlighted the need to support optional travel services, including:

- Extra baggage
- Preferred seat selection
- Special-needs assistance
- Pet travel

Each service must follow airline rules and capacity limits.

6. Types of System Users:

Customers requested support for multiple user roles with different permissions:

- **Traveler (User):** Search, book, pay, modify, cancel, manage bookings, add services
- **Support/Operations Staff:** Handle modifications, cancellations, refunds, and support tickets

7. Airline Rules, Policies, and Restrictions:

Customers stressed the importance of displaying airline policies clearly for:

- Baggage allowance
- Modifications
- Cancellation rules
- Refund calculations

These policies must be enforced automatically during booking, modification, or cancellation.

8. System Reliability, Performance, and Capacity:

Customers confirmed expected business capacity:

- 6,000–8,000 total registered users
- 10,000–15,000 monthly flight searches
- 250–450 monthly bookings

The system must be reliable and fast enough to handle peak periods.

A.2 Minutes of Meetings for the Team

Throughout the project, the team maintained structured and regular internal meetings to ensure alignment, accountability, and steady progress. On average, the team conducted **two meetings per week**, with each meeting lasting between **30 minutes and up to two hours**, depending on the agenda and project phase.

These meetings were primarily used for team members to present and demonstrate their ongoing work, ask questions, and discuss challenges. Each member was given the opportunity to explain their progress, clarify uncertainties, and receive feedback from the rest of the team. This approach helped identify issues early and ensured shared understanding of system design and implementation decisions.

The **Project Manager** played a central role in these meetings by reviewing the presented work, providing constructive feedback, and assigning follow-up tasks. At the end of each meeting, responsibilities were clearly defined, and each team member was informed of the specific tasks to be completed before the next session.

In parallel, the **Requirement Engineer** was responsible for reviewing the outputs of all tasks to ensure that they remained consistent with the documented user and system requirements. This included verifying that design decisions, diagrams, and implementation work aligned with the customer's expectations and business needs. Any deviations or ambiguities were highlighted and discussed during meetings so that corrective actions could be taken promptly.

As the project progressed into more demanding stages, the same meeting structure was maintained to support implementation and integration activities. These regular sessions enabled continuous feedback, effective coordination, and ensured that all work delivered by the team remained aligned with customer requirements and overall project goals.