CALIFORNIA STATE UNIVERSITY
LONG BEACH
College of Business

Last Chance Air – Last-Minute Flight Deals Platform

IS 699: Information Systems Project
Fall 2025
Professor: Dr. Abraham Asher

By:

Lokesh Dhamodharan - 033021311

Vijay Krishna Ramraj- 033696466

Kalyan Uppalapatti – 033693385

Table of Contents

Table of Figures

1. Executive Summary

This report presents the design, development, and evaluation of **LastChance Air**, a full-stack web application implemented as the final project for IS 699. The system simulates a real-world last-minute flight booking platform designed to help users quickly identify discounted airfares for domestic U.S. travel. The platform addresses the dual challenge faced by airlines and urgent travelers: airlines routinely operate flights with unfilled seats close to departure, while travelers needing immediate trips struggle to find affordable options on large, complex travel sites.

Unsold seats represent lost revenue for airlines, and the absence of a dedicated solution forces travelers to spend time sifting through information not optimized for urgent travel. In response to this problem, our project provides an integrated platform that surfaces last-minute flight deals by applying dynamic discounting logic and a ranking model inspired by airline revenue-management techniques.

The system uses a scoring algorithm that evaluates each flight based on factors such as proximity to departure, current seat availability, and discount percentage. By computing a weighted score, the system highlights the most attractive last-minute opportunities for users in a clear and intuitive interface. This mimics real airline practices in which urgency, occupancy, and price adjustments determine fare visibility and promotion priority.

The application is implemented as a complete web-based system with three primary components:

1. A **backend engine** built with Node.js and Express that processes search queries, applies discount logic, and manages user bookings.
2. A **relational database layer** (SQLite) that stores users, flights, seats, bookings, and discount rules, ensuring consistency and integrity throughout the booking workflow.
3. A **frontend interface** developed with HTML, CSS, and JavaScript that enables users to search flights, view ranked deals, select seats, choose add-ons, and confirm bookings.

The system replicates essential elements of commercial travel platforms, including real-time price adjustments, dynamic seat availability, interactive seat maps, add-on services, and automated booking confirmations sent by email.

The project followed the complete information systems life cycle: opportunity identification, feasibility assessment, project planning, modeling using UML and ER diagrams, system development, implementation, documentation, testing, risk analysis, and evaluation. This report documents each phase in detail, demonstrating how LastChance Air serves both as a practical solution to a real industry problem and as a comprehensive capstone project aligned with the goals of IS 699.

2. Identification / Selection of Project / Opportunity Evaluation

2.a Background

Airlines frequently operate flights with empty seats, especially during the final days leading up to departure. Because aircraft depart regardless of occupancy, any unfilled seats represent **direct and irreversible revenue loss**. At the same time, travelers who need to book urgent or last-minute trips often encounter **high prices, limited transparency, and complex booking platforms** that are not optimized for fast decision-making.

Existing travel aggregators such as Expedia, Kayak, and Skyscanner are designed primarily for **general travel planning**, offering extensive filters, long-term search ranges, and flexible itinerary support. However, these platforms do **not specialize in last-minute travel**, nor do they highlight deals created specifically due to expiring inventory. As a result, there exists a clear gap between **airline operational inefficiencies** and **user needs** during urgent travel scenarios.

Users who want a quick, affordable booking experience must sift through large volumes of irrelevant information, while airlines lack a simple way to strategically surface discounted seats that would otherwise remain unsold. Solving this requires a system that can simulate airline pricing logic while presenting a simple, intuitive user experience.
Our team selected **LastChance Air** as the project topic because it directly addresses this real-world inefficiency, is technically feasible within course constraints, and provides an excellent opportunity to demonstrate full-stack system development, modeling, and project management principles aligned with the goals of IS 699.

2.b Strategic Planning

Mission

To design and develop a last-minute flight booking platform that identifies discounted near-term flights and presents users with a fast, intuitive, and deal-focused booking experience.

Objectives

1. Implement a realistic flight search engine for domestic U.S. routes.
2. Develop a dynamic discounting and ranking model that highlights the best last-minute deals.
3. Provide an intuitive frontend interface that enables users to search, view deals, select seats, and confirm bookings quickly.
4. Integrate seat selection, add-on services, and booking confirmation emails to simulate

real airline workflows.

5. Apply formal information systems project management practices, including planning, feasibility analysis, risk management, version control, and documentation.

Competitive Strategies

- Focus exclusively on **near-term departures (3–7 days)** rather than general travel planning.
- Apply **dynamic discount logic** based on seat availability, departure urgency, and value scoring.
- Provide a **clean and lightweight UI** designed for users in a hurry.
- Implement realistic features such as **seat maps, add-ons, booking confirmation emails**, and a scoring model for ranking deals.
- Structure the system using **separation of concerns**—frontend, backend, database, and pricing engine—to support maintainability and future enhancements.

2.c Constraints

The project was subject to several constraints:

- **Time Constraint:** Development had to be completed within a single academic semester.
- **Data Constraint:** Real airline APIs were not used due to cost and complexity; instead, synthetic flight data was generated.
- **Operational Constraint:** No real payment gateway was integrated; booking flows are simulated.
- **Technical Constraint:** SQLite was used instead of enterprise-level databases due to resource limitations.
- **Scope Constraint:** The system is limited to **domestic U.S. flights** and focuses only on last-minute travel scenarios.

2.d Overall System Need

A variety of users can benefit from a platform that simplifies last-minute flight discovery:

- Travelers needing urgent trips who want affordable fares quickly.
- Students, families, and business travelers looking for last-minute opportunities.
- Users frustrated with the complexity of major travel platforms during urgent booking.
- Airline operations (simulated) aiming to reduce revenue loss from unsold seats.

To be useful, such a system must be **fast**, **simple**, and **focused only on last-minute deals**. LastChance Air meets these needs through:

- A ranking model that prioritizes flights based on urgency and value.
- A simple UI designed for quick decision-making.
- A complete booking workflow that mirrors real airline processes, including seat selection and confirmation emails.

2.e Strategic Fit

LastChance Air aligns strongly with the goals of IS 699 and the broader Information Systems curriculum. The project integrates:

- Systems analysis and design principles
- Project initiation, planning, and feasibility assessment
- Frontend and backend software development
- Database modeling and ER design
- UML diagrams including use case, class, sequence, and activity models
- Risk management, scheduling, and documentation
- User interface design and implementation

The project also reflects current industry practices in **dynamic pricing**, **inventory optimization**, and **customer experience design**, demonstrating both technical and managerial competencies expected in real-world information systems applications.

3. Initiation and Planning of System Project

3.a Charter

Purpose:
To design, build, and evaluate a working prototype of **LastChance Air**, a full-stack web application that simulates a last-minute domestic flight booking system. The purpose of the platform is to address the operational inefficiency airlines face due to unsold seats close to departure and to help travelers find affordable, urgent travel options through dynamic pricing and simplified booking workflows.

Scope:
The scope of this project includes the following:

- Creation of synthetic flight data to simulate airline inventory.
- Design and implementation of a dynamic last-minute discounting and ranking engine.
- Development of backend REST API endpoints using Node.js and Express.js.
- Development of a frontend UI using HTML, CSS, and JavaScript to support flight search, deal display, seat selection, and booking confirmation.
- Database design and integration using SQLite to store users, flights, bookings, and seats.
- Implementation of an email confirmation service using Nodemailer.
- System integration, testing, documentation, presentation, and delivery of all required artifacts.

Team Responsibilities:

- **Vijay Krishna Ramraj** – Frontend development, UI logic, dynamic deal rendering, documentation.
- **Lokesh Dhamodharan** – Backend development, API design, pricing engine, database integration.
- **Kalyan Uppalapatti** – Seat map implementation, booking workflow, system modeling (UML + ERD), project coordination.

3.b Statement of Work

The team will:

- Define project requirements and constraints.
- Design and document the system architecture and UML models.
- Generate synthetic flight data and configure the ranking/discount logic.
- Implement backend services (search, booking, confirmation).
- Develop the frontend interface for search, deal display, and user workflow.
- Integrate, test, and validate the complete booking flow.
- Deliver all final project artifacts including the written report, diagrams, Gantt chart, slide presentation, and live demo.

Deliverables:

- Complete frontend and backend source code.
- SQLite database schema and integration scripts.
- UML Diagrams: use case, class, activity, sequence, state, and ERD.
- Gantt chart and project management documentation.
- Fully functional prototype demonstrating booking flow with discount logic.
- Final written report and oral presentation.

3.c Introduction

3.c.i Project Overview

LastChance Air is a browser-based application that enables users to quickly search for last-minute domestic flights. The user interacts with a simple search interface to enter travel details. The backend processes the query, applies dynamic discount and ranking logic, and returns a list of deals sorted by urgency and value. Users can then select a specific flight, view seat availability, choose add-ons such as meals or extra legroom, and confirm the booking. An automated confirmation email is generated using Nodemailer.
The system replicates the essential steps of real-world airline booking applications while remaining lightweight and optimized for educational use.

3.c.ii Recommendation

We recommend continuing development in the following areas:

- Integration with real airline APIs (e.g., Amadeus) to replace synthetic data.
- Deployment of the system to a public cloud platform for wider accessibility.
- Addition of advanced pricing strategies and machine-learning-based demand prediction.
- Support for round-trip searches and multi-city itineraries.
- Add-ons such as user accounts, saved searches, and loyalty points.

3.d System Description

3.d.i Alternatives

The team considered several technical alternatives:

1. Using a full frontend framework such as React or Angular for UI development — rejected due to time constraints and higher learning curve for new contributors.
2. Using a more complex relational database such as PostgreSQL — avoided in favor of SQLite for simplicity and portability.
3. Integrating real-time airline APIs not feasible due to cost, authentication requirements, and API rate limits.

3.d.ii Recommendation

The team selected a lightweight stack consisting of:

- **Frontend:** HTML, CSS, and JavaScript
- **Backend:** Node.js + Express.js
- **Database:** SQLite

This combination ensured fast development cycles, easy debugging, and sufficient functionality to model real airline workflows within the course timeline.

3.e Feasibility Assessment

3.e.i Economic Feasibility

The project uses open-source technologies (Node.js, SQLite, JavaScript, GitHub), resulting in negligible development cost. No licensing fees or cloud-hosting expenses were required.

3.e.ii Technical Feasibility

The team had prior experience with full-stack development, REST APIs, and relational databases. The chosen technologies supported smooth integration of features such as discount logic, seat maps, and email confirmation.

3.e.iii Operational Feasibility

The interface is simple, requires no training, and supports intuitive user workflows. Operations follow a familiar travel-booking layout, making the system easy to adopt and understand.

3.e.iv Schedule Feasibility

Development was organized into iterative sprints using Azure DevOps. Milestones included search functionality, ranking engine, seat map, booking flow, email confirmation, and UI refinement. Despite the ambitious scope, all core features were completed on schedule..

3.f Project Plan

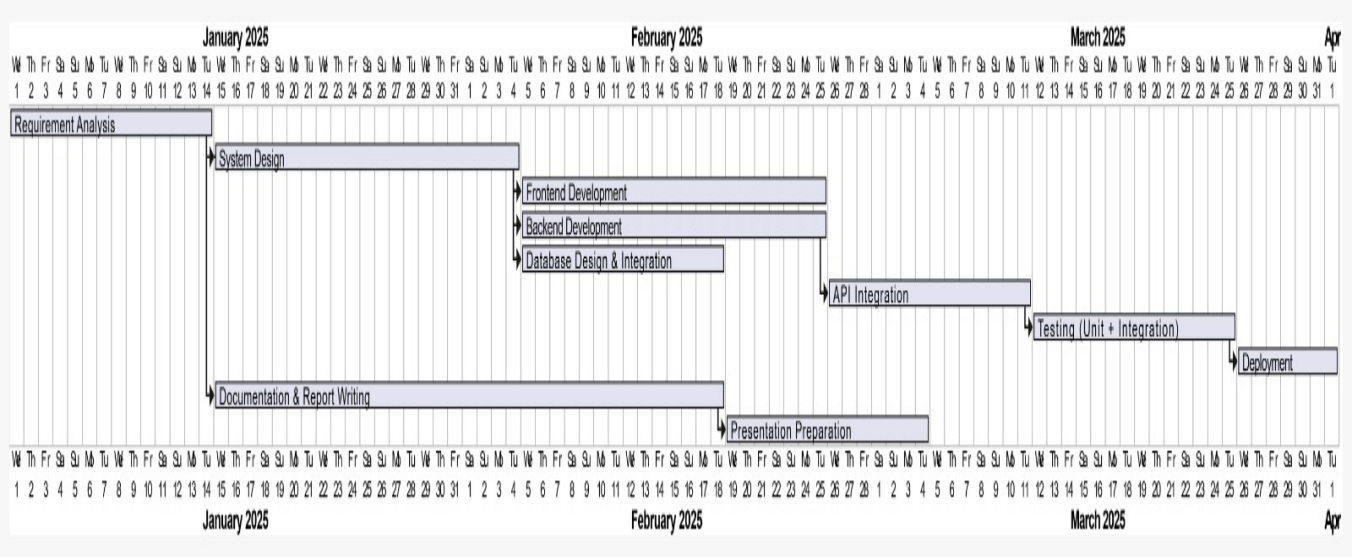
3.f.i Implementation Schedule – Gantt Chart

The project followed a detailed implementation timeline from September 10, 2025 to December 11, 2025. Major phases included:

- Initiation – define goals, select project, create charter.
- Requirements & Modeling – UML diagrams, ERD, system architecture.
- Backend Development – APIs, ranking logic, database integration.
- Frontend Development – search UI, results, seat map, booking pages.
- Integration & Testing – end-to-end flow validation, bug resolution.
- Deployment & Closure – documentation, final report, and presentation.

The Gantt chart reflects dependencies such as the booking workflow depending on both seat map completion and backend storage logic.

.



The Gantt chart also reflects task assignments and predecessors. For example, model evaluation depended on model building tasks, and UI deployment depended on both backend implementation and model availability.

3.g Communication Plan

The team maintained consistent communication using the following tools:

- Weekly scheduled meetings via Microsoft Teams.
- Daily or ad-hoc updates using Teams chat.
- Task tracking and sprint planning in Azure DevOps.
- GitHub pull requests for version control and code review.

This ensured transparency and minimized misunderstandings during development.

3.h Quality Management Plan

Key quality practices included:

- Use of GitHub version control to avoid code conflicts.
- Code reviews before merging new features into the main branch.
- Validation of discount logic, search accuracy, booking correctness, and email functionality.
- Manual testing of the UI after each feature update.
- Consistency checks for database integrity and seat availability.

3.i Change Control Plan

The team followed a structured but lightweight change-control workflow:

1. A proposed change was documented as a work item in Azure DevOps.
2. The team discussed feasibility during meetings or chat.
3. Approved changes were added to the current or next sprint.
4. Changes were implemented in a separate GitHub branch.
5. Another team member reviewed the pull request before merging.

This prevented scope creep and ensured stability across all system components.

3.j Risk Plan

3.j.i Project Risks

Risk	Likelihood	Consequence	Mitigation
Delay in building discount logic	Medium	High	Early prototyping and iterative testing
Integration issues (frontend + backend + DB)	Medium	Medium	Frequent integration tests
Time constraints	High	Medium	Feature prioritization and phased development

3.j.ii System Risks

Risk	Likelihood	Consequence	Mitigation
Incorrect price calculation	Medium	High	Unit tests for ranking & discount logic
Data inconsistency (seat availability)	Low	High	DB constraints + backend validation
Email delivery issues	Medium	Medium	Retry logic + SMTP fallback templates

3.k Microsoft Teams

Pros / Advantages

- Central communication hub for chat, meetings, and file sharing.
- Meeting recordings for absent members.
- Integration with Office tools for collaborative editing.

Cons / Disadvantages

- Older messages difficult to search.
- High notification volume may cause distractions.

Potential Usage / Lessons Learned

- Create dedicated channels for architecture, backend, frontend, and documentation.
- Pin important messages to maintain structure.

Overall Assessment

Microsoft Teams was effective for collaboration and helped keep the team aligned throughout the project.

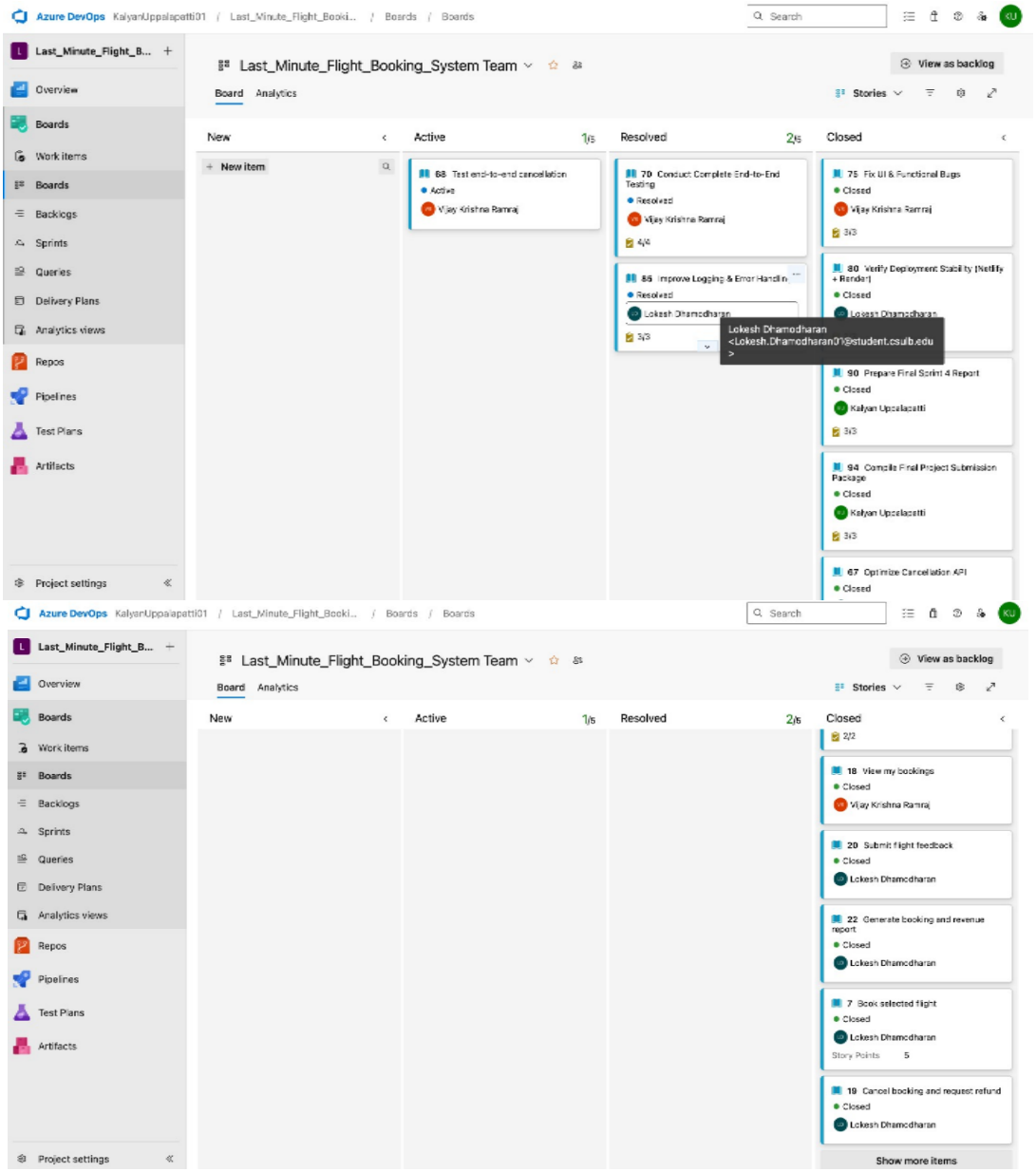
3.l Azure DevOps

Link

Devops

How the Team Used Azure DevOps

- Created Epics for backend, frontend, database, and pricing engine.
- Decomposed tasks into user stories.
- Planned one- to two-week sprints.
- Used Boards to track progress.
- Recorded sprint reviews and retrospectives.



Pros / Advantages

- Excellent visibility into project progress.
- Balanced workload distribution.
- Served as a historical log of decisions and changes.

Cons / Disadvantages

- Initial learning curve.
- Requires disciplined status updates.

Potential Usage / Lessons Learned

In a future project, the team would:

- Use more granular tasks.
- Apply tags (frontend, backend, logic) for easier filtering.
- Leverage built-in analytics for retrospectives.

Overall Assessment

Azure DevOps proved to be a valuable management tool that improved organization, transparency, and accountability across the project.

4. System Requirements

4.a Approach to Identifying Requirements

System requirements for **LastChance Air** were identified using a structured and iterative approach. The team conducted brainstorming sessions, evaluated user needs, analyzed the workflow of real airline booking systems, and reviewed existing travel platforms to understand common functionalities and limitations. Use-case modeling and feasibility assessments further refined the requirements, ensuring they aligned with project goals, technical constraints, and the intended user experience.

The focus was on defining a practical, minimal, and functional solution that supports last-minute flight search, discount ranking, seat selection, booking confirmation, and system interaction through a clean and intuitive interface.

4.b Functional Requirements

1. The system must allow users to search for flights by entering origin, destination, and departure date.
2. The system must apply dynamic discounting and ranking logic to highlight last-minute deals.
3. The system must allow users to select seats using an interactive seat map.
4. The system must allow users to confirm a booking and store it in the SQLite database.
5. The system must send an email confirmation after a successful booking.
6. The system must allow users to view and manage their bookings under the “My Bookings” section.

Three of these requirements Flight Search, Seat Selection, and Booking Management—will be expanded in the modeling section through full lifecycle diagrams including use case, activity, sequence, and state mod

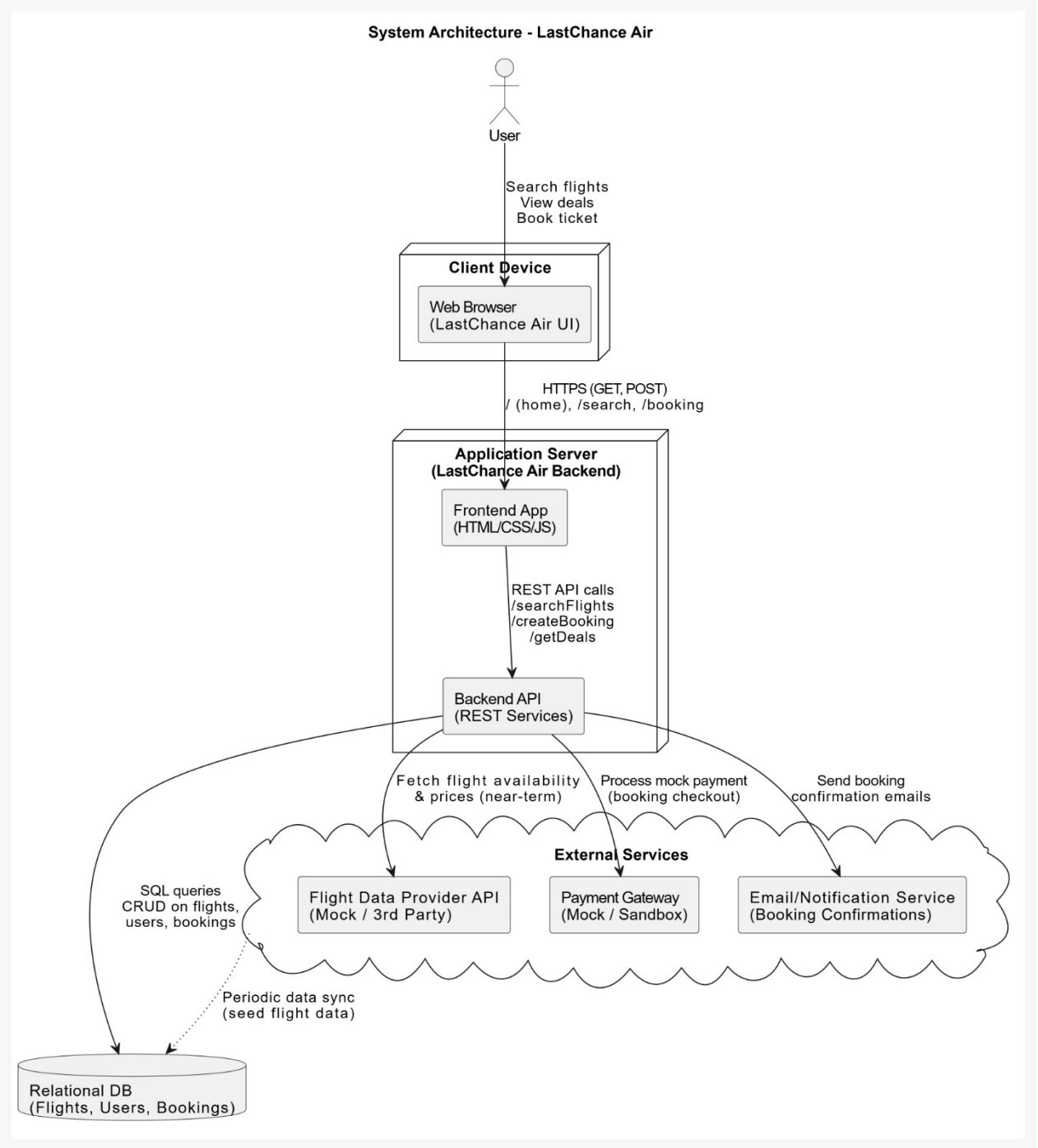
4.c Non-Functional Requirements

1. **Usability:**
   The system must provide a simple, clean, intuitive user interface suitable for users making urgent last-minute bookings.
2. **Performance:**
   Flight search results and discount ranking must complete within **2–3 seconds** under normal load.
3. **Security:**
   User passwords must be hashed; all booking and search operations must be validated on the backend.
4. **Reliability:**
   The system must ensure booking data, seat availability, and user information remain consistent and protected.
5. **Responsiveness:**
   The system must behave consistently on modern desktop and mobile browsers.

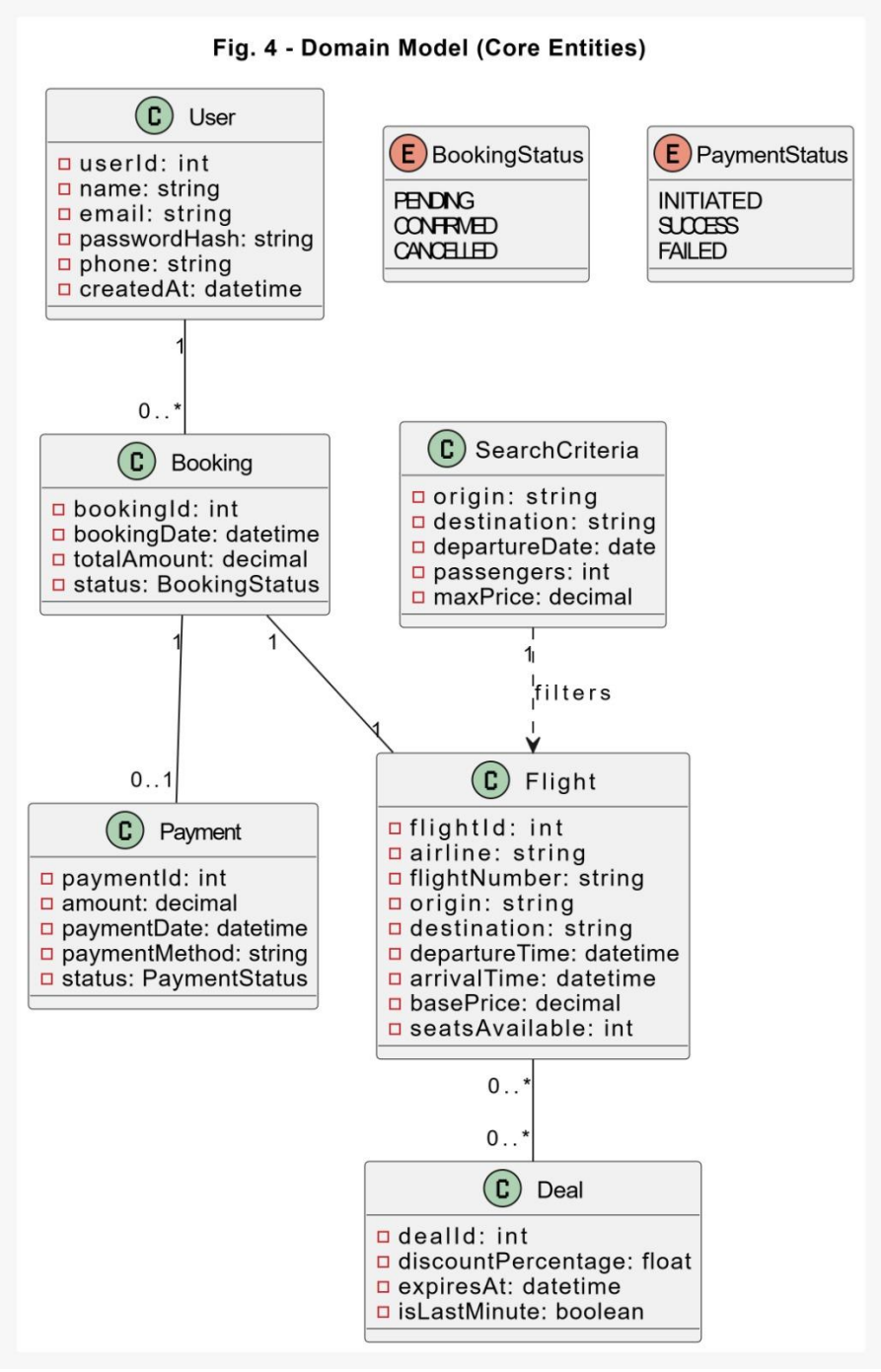
These non-functional requirements were addressed through careful UI design, efficient backend filtering and ranking logic, database constraints, and lightweight implementation choices suitable for rapid academic development

5. Models

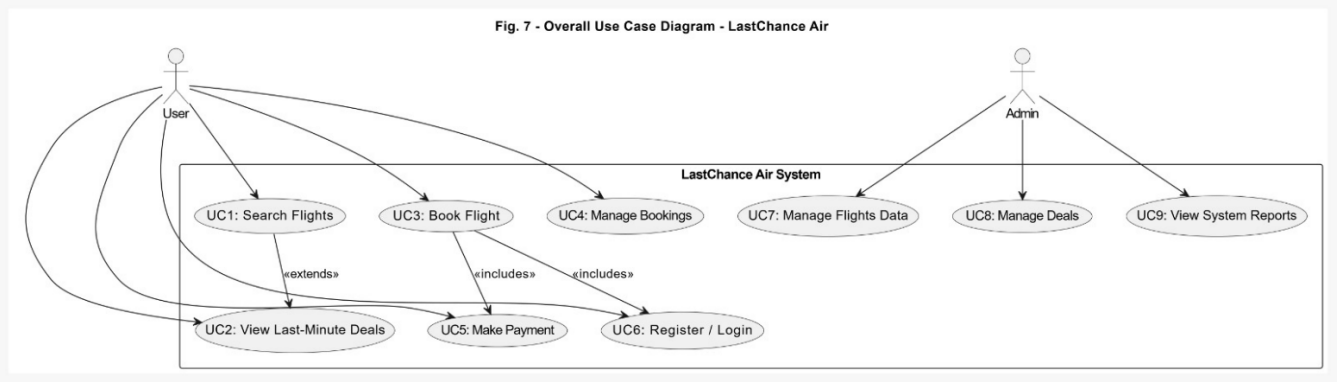
System Architecture Model



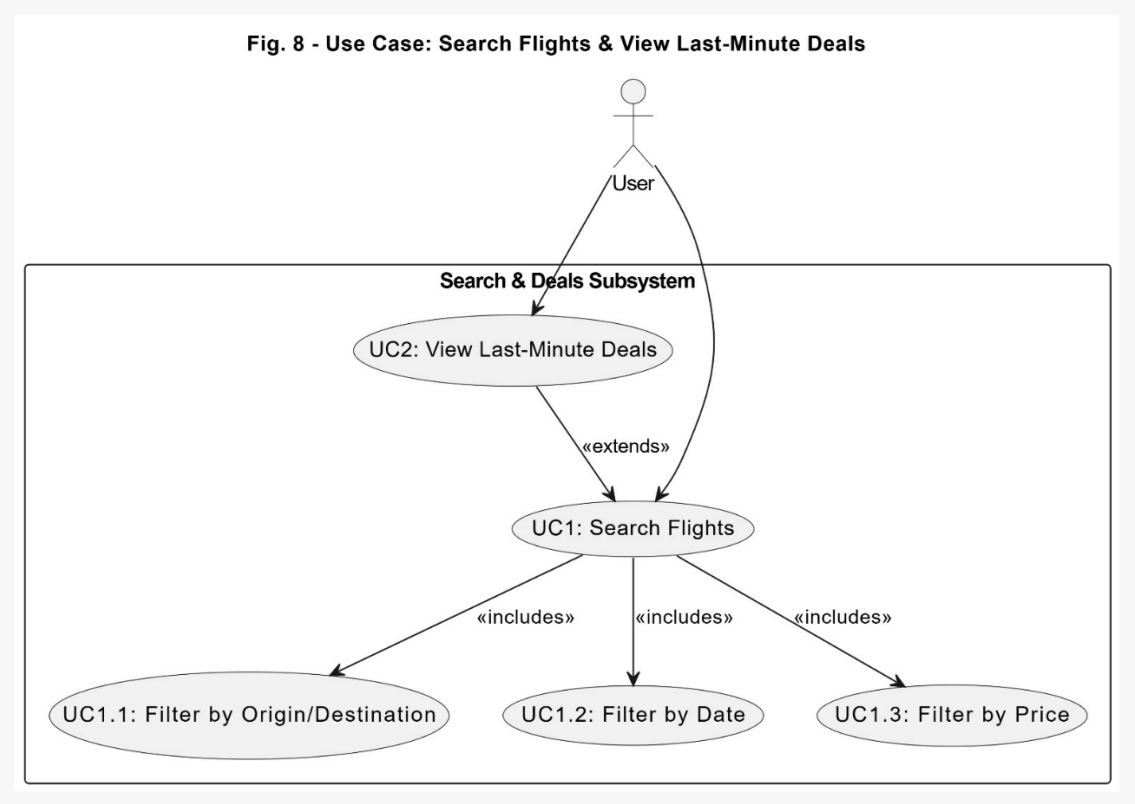
Class Diagrams



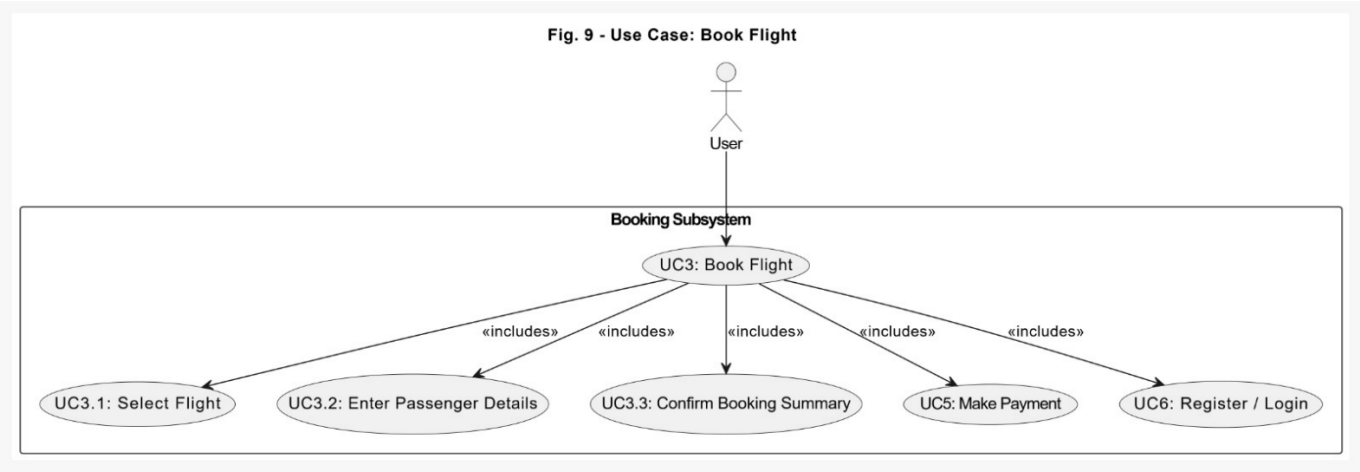
Use Case Diagrams



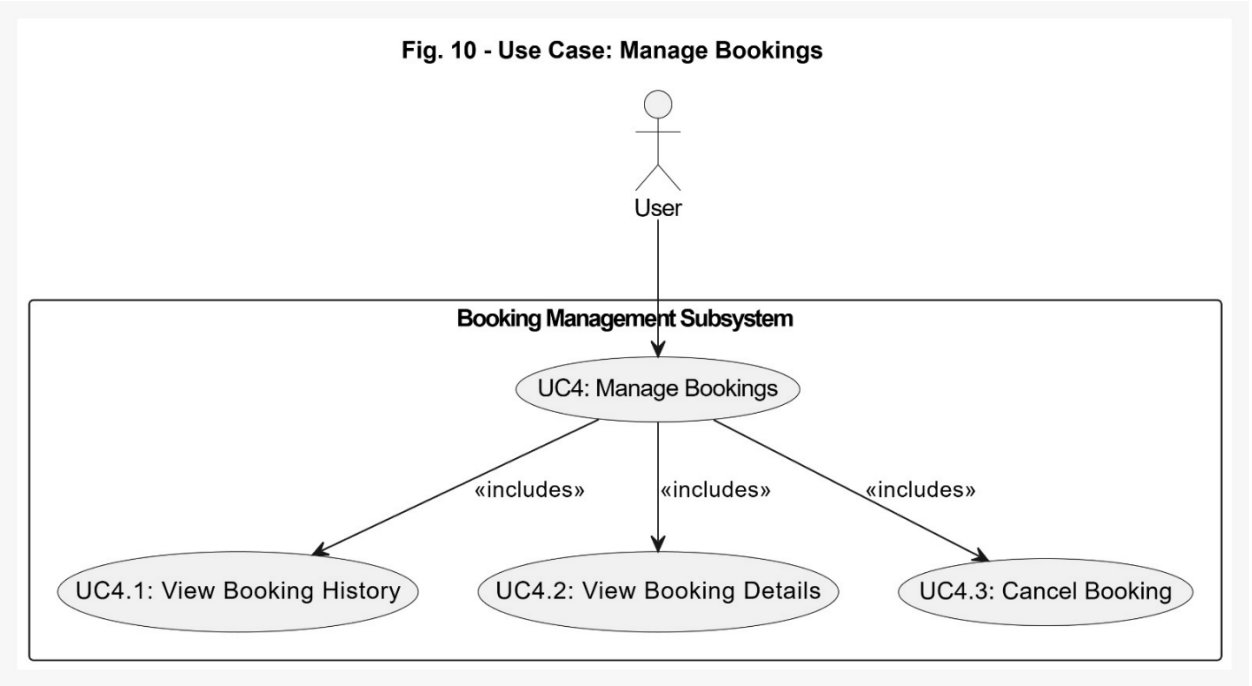
USE CASE DIAGRAM 1 - System Overview



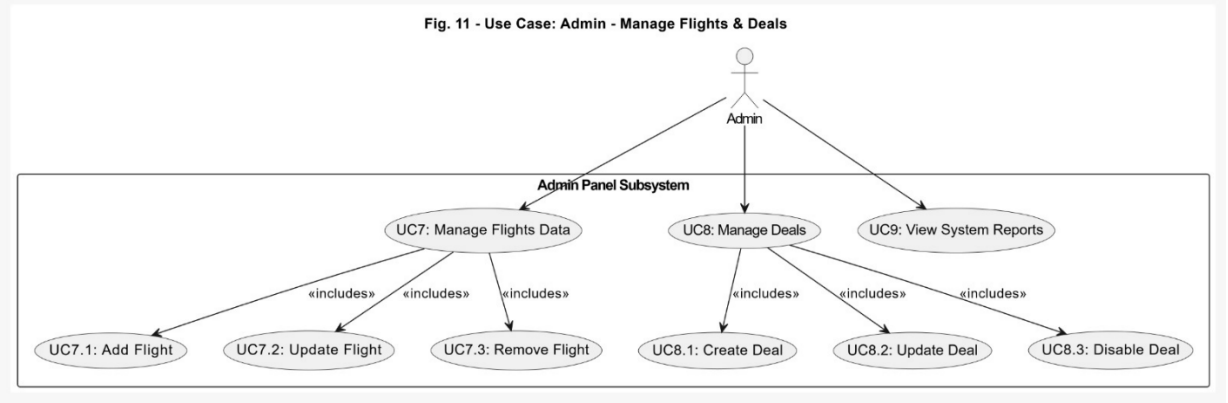
USE CASE DIAGRAM 2 – Use Case Search Flights And view Last Minute Deals



USE CASE DIAGRAM 3 — Book Flight

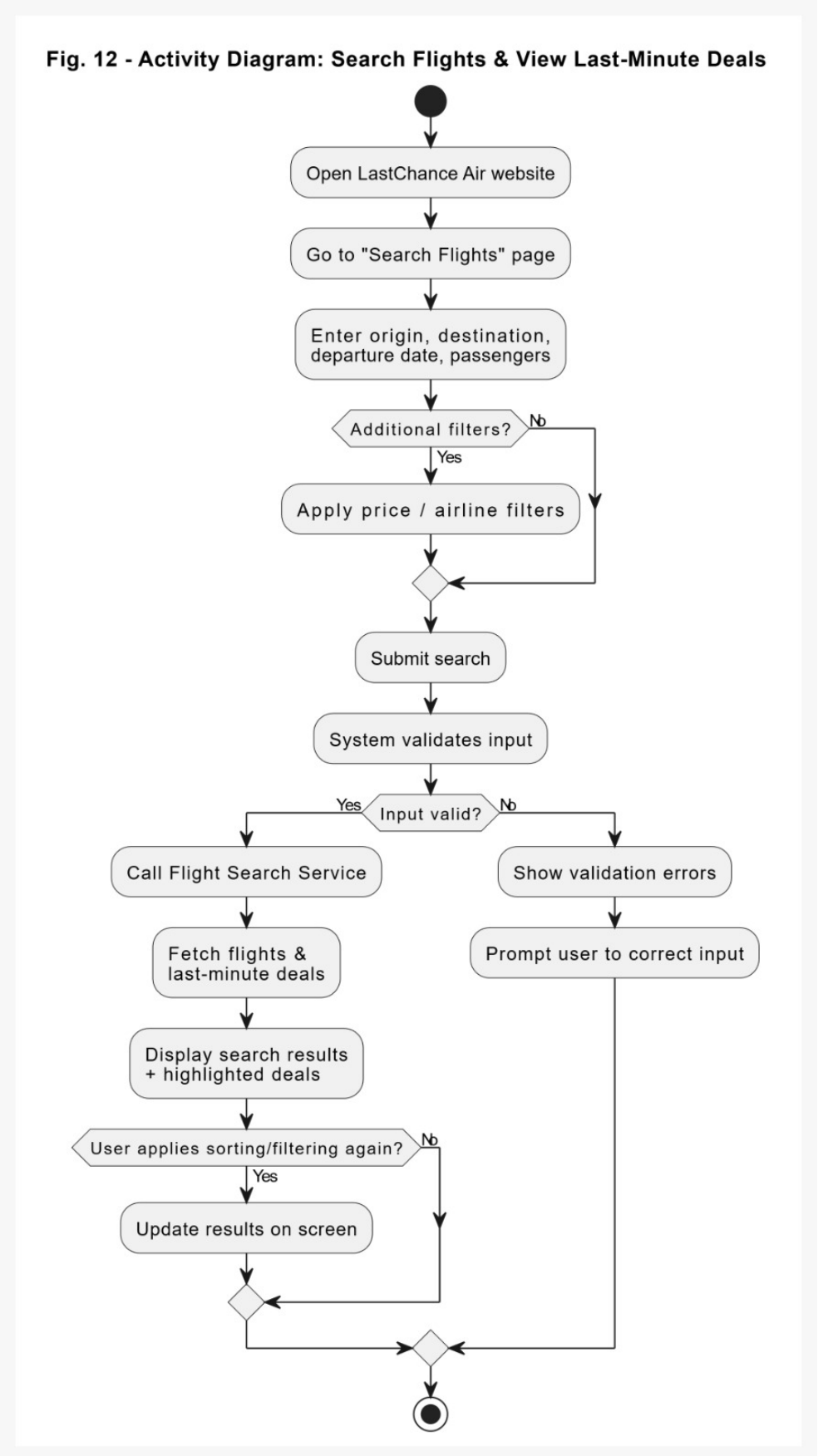


USE CASE DIAGRAM 4 — Manage Booking

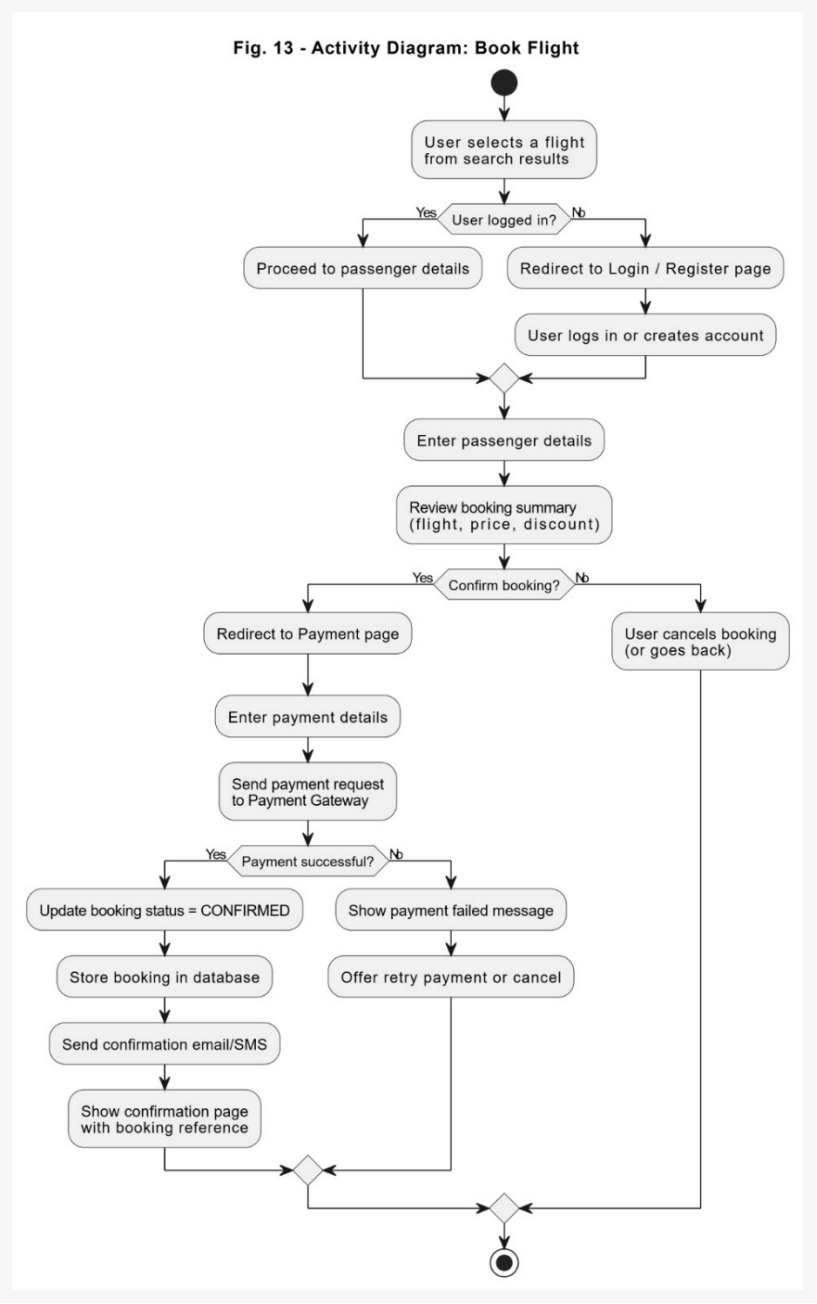


USE CASE DIAGRAM 5 — Manage Booking

Activity Diagram

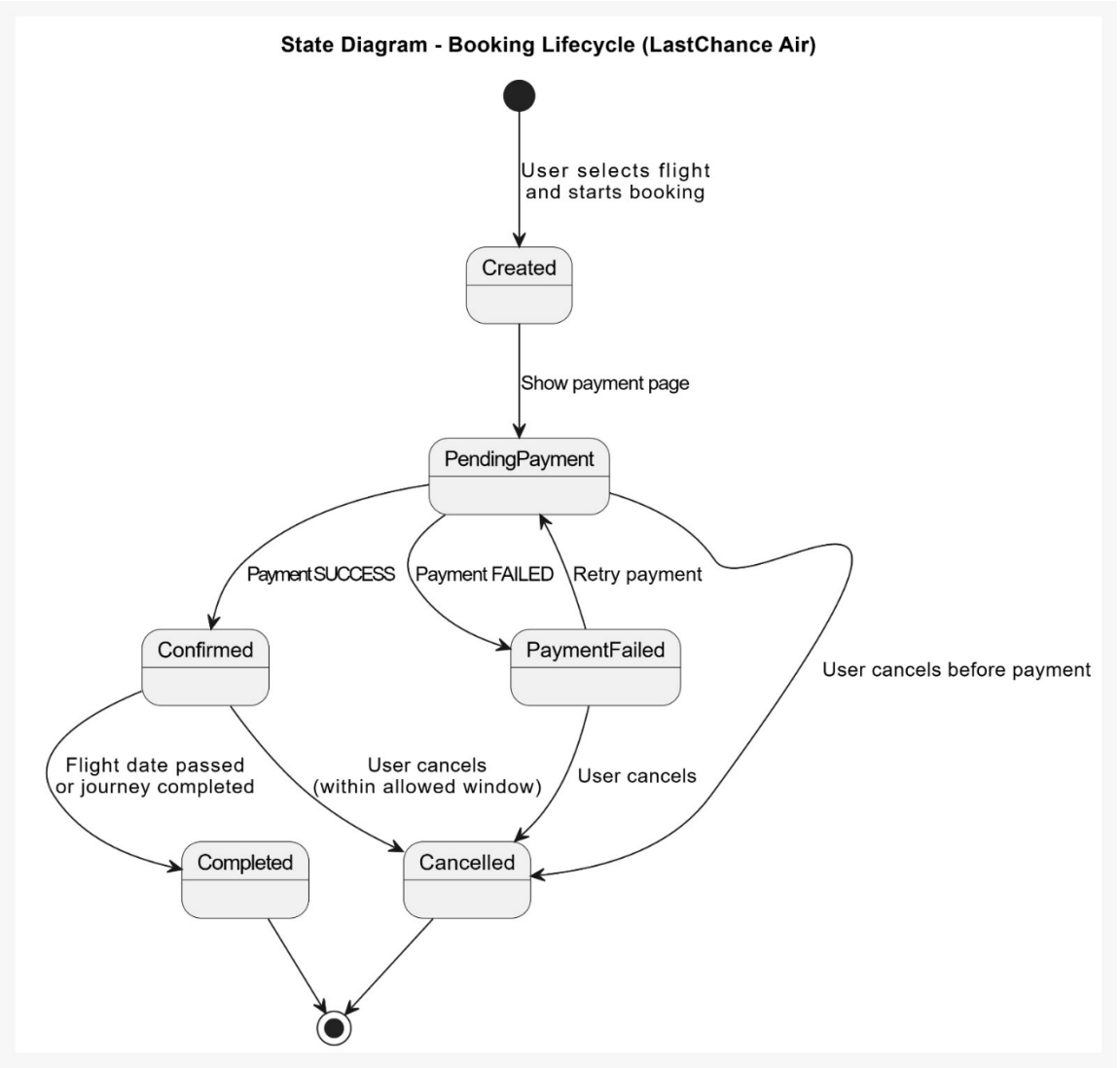


ACTIVITY DIAGRAM 1 — Search Flights And view last Minute deals



ACTIVITY DIAGRAM 2 — Book Flight

State Diagram



Sequence Diagram

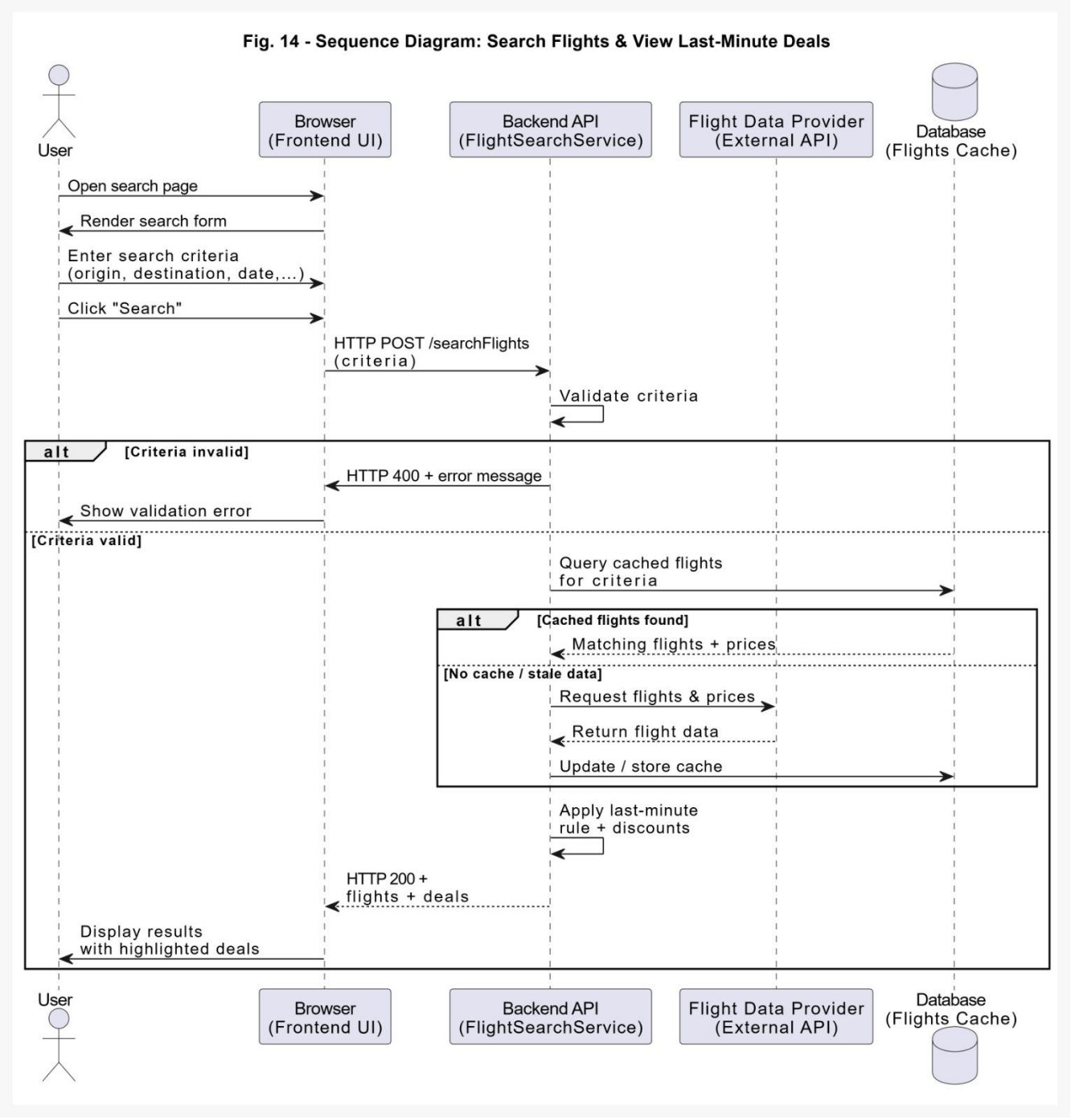
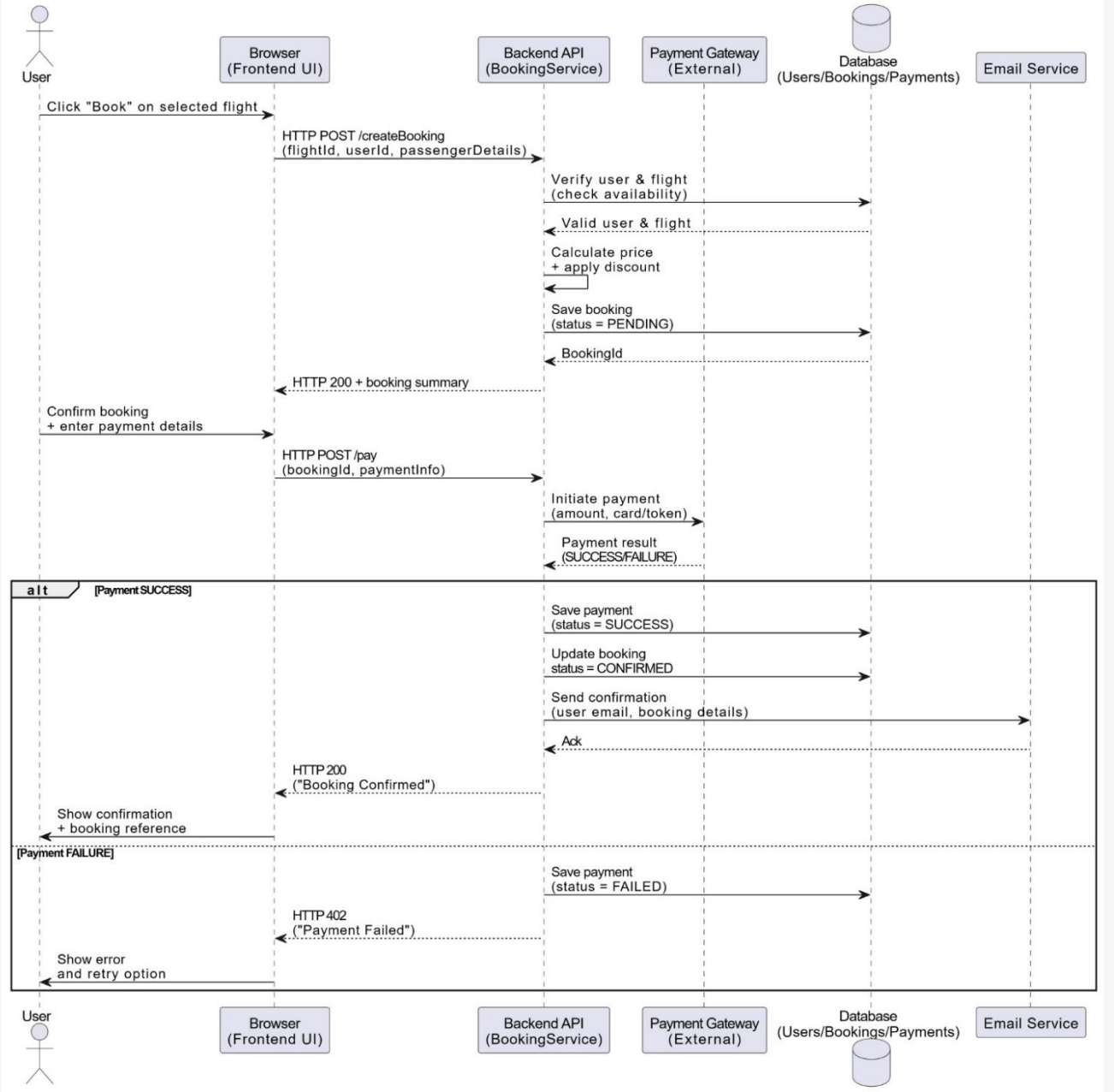
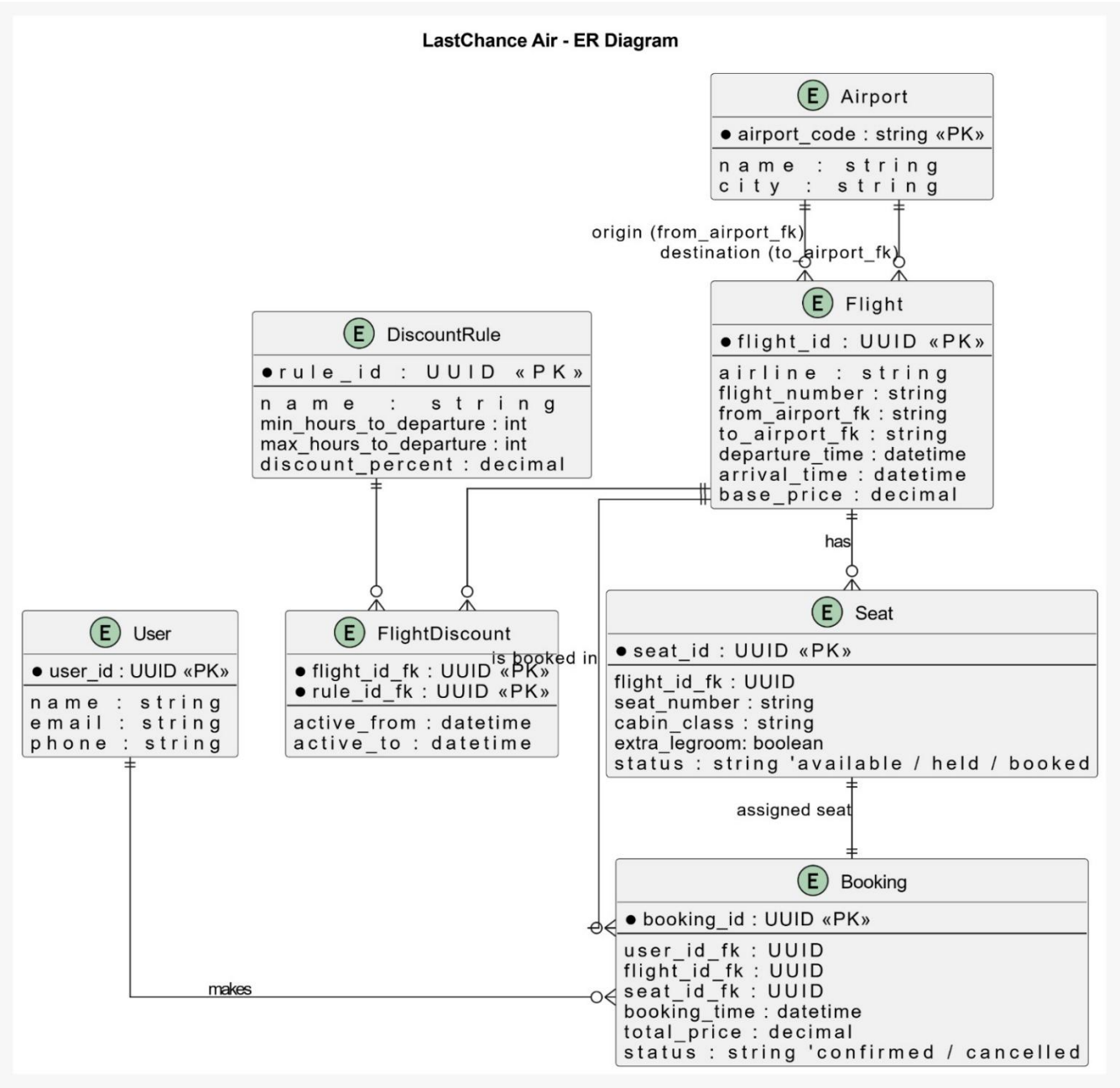


Fig. 15 - Sequence Diagram: Book Flight & Payment



ER Diagram


LastChance Air - ER Diagram
Airport
● airport_code : string «PK»
name : string
city : string
origin (from_airport_fk)
destination (to_airport_fk)
Flight
● flight_id : UUID «PK»
airline : string
flight_number : string
from_airport_fk : string
to_airport_fk : string
departure_time : datetime
arrival_time : datetime
base_price : decimal
DiscountRule
● rule_id : UUID «PK»
name : string
min_hours_to_departure : int
max_hours_to_departure : int
discount_percent : decimal
has
Seat
● seat_id : UUID «PK»
flight_id_fk : UUID
seat_number : string
cabin_class : string
extra_legroom: boolean
status : string 'available / held / booked
User
● user_id : UUID «PK»
name : string
email : string
phone : string
FlightDiscount
● flight_id_fk : UUID «PK»
● rule_id_fk : UUID «PK»
active_from : datetime
active_to : datetime
is booked in
assigned seat
Booking
● booking_id : UUID «PK»
user_id_fk : UUID
flight_id_fk : UUID
seat_id_fk : UUID
booking_time : datetime
total_price : decimal
status : string 'confirmed / cancelled
makes

6. System Development

6.a Hardware Requirements

- Development machine with at least **8–16 GB RAM** and a modern multi-core CPU.
- For backend + frontend development, no GPU is required.
- Minimum **5–10 GB** of free disk space to store code, SQLite database files, and test data.
- A stable internet connection for documentation, package installation, and dependency updates.
-

6.b Software Requirements

- Operating System: Windows, macOS, or Linux
- **Backend:** Node.js (v16+) + Express.js framework
- **Database:** SQLite3 (file-based relational database)
- **Frontend:** HTML5, CSS3, JavaScript
- Additional Libraries:
- bcrypt.js for password hashing
- nodemailer for email confirmation
- uuid for unique booking ID generation
- dotenv for environment variable configuration
- Development Tools:
- Visual Studio Code
- Postman or Thunder Client for API testing
- Git and GitHub for version control

6.c Version Control Systems

GitHub was used to store and manage all project files, including:

- Frontend UI components
- Backend server code
- Database schema and sample data
- Documentation and UML diagrams
- Environment configuration templates

Git workflows ensured:

- Version tracking
- Parallel development through branches
- Code review before merging
- Ability to revert changes when necessary

6.d Cloud Usage

The system was developed and tested locally. Due to resource and budget limitations, cloud deployment was not performed during the course timeline.
However, the system is prepared for future deployment on services such as:

- Azure App Service
- Netlify (frontend) + Render/Railway (backend)
- AWS Elastic Beanstalk

The architecture supports containerization via Docker for production-ready cloud deployment.

6.e Development Approach

The team followed a **Rapid Application Development (RAD)** approach with Agile principles:

- Early prototypes of search, results page, and booking screens
- Frequent iteration based on user flow testing
- Backend and frontend were developed in parallel
- User stories and sprint tasks managed in Azure DevOps
- Continuous refinement of features such as ranking logic, seat map, and email confirmation

This approach minimized rework and enabled steady progress throughout the semester.

6.f New Developer Instructions:

A new developer joining the **LastChance Air** project should:

- Clone the GitHub repository.
- Install Node.js and SQLite on their machine.
- Run npm install to install dependencies.
- Create a .env file containing:
- SMTP credentials
- Port number
- Start the backend server using:
  node server.js
- Open index.html in the browser for the frontend.
- Review architecture diagrams, sequence flows, and booking logic in documentation.
- Test core workflows: search → ranking → seat selection → booking → email.

7. Implementation

7.a Testing

Functional Testing:
- Flight search
- Ranking and discount logic
- Seat availability updates
- Booking creation and cancellation

Integration Testing:
- Frontend ↔ Backend communication
- Backend ↔ SQLite database
- Booking flow ↔ Email confirmation

UI Testing:
- Page navigation
- Form validation
- Responsive behavior

Edge Case Testing:
- Invalid dates
- Unavailable seats
- Duplicate bookings
- Incorrect email formats

7.b Deployment

The system currently runs locally:
- Backend runs on a local Node.js server
- Frontend is served directly from browser
- SQLite database stored in local project directory

Future deployment options include:
- Netlify + Render
- Azure App Service
- AWS Elastic Beanstalk

7.c Training

Unlike deep-learning projects, LastChance Air does not require ML model training.
Instead, "training" relates to:
- Calibrating discount rules
- Testing value-scoring logic for ranking
- Validating seat availability behaviors
- Simulating pricing across different flight patterns

These were refined through iterative testing.

7.d Support

Support for the system includes:
- Updating npm dependencies

- Validating email delivery (SMTP credentials may expire)
- Database backup and maintenance
- Fixing UI issues based on feedback
- Ensuring compatibility with updated browsers

8. System Documentation

8.a Forms / User Interface

1. Search Screen

Purpose: Allow users to enter origin, destination, and date.
Inputs: Origin, destination, departure date.
Output: List of available flights ranked by discount and urgency.

2. Deal Results Screen

Purpose: Display dynamically ranked flight deals.
Inputs: Filtered results from backend.
Output: Sorted list with prices, discounts, and availability.

3. Seat Selection & Add-ons

Purpose: Allow user to select a seat and optional extras.
Inputs: Seat choice, add-on selections.
Output: Updated total price and confirmation prompt.

4. Booking Confirmation Screen

Purpose: Display booking details and send confirmation email.
Inputs: Passenger details.
Output: Confirmation message, email receipt.

8.b Reports / User Interface

1. My Bookings Report

Displays:

- Flight route
- Seat selected
- Total price
- Status (Confirmed / Cancelled)
- Booking timestamp

2. Suggested Deals Report

Displays:

- Top-ranked last-minute deals
- Discount labels
- Seats remaining
- Final discounted prices

9. Application State / Developed Features / To-Do / Prototype

9.a Completed Features

- Flight search with filters
- Dynamic discount calculation
- Ranking model for last-minute deals
- Seat map selector
- Booking creation and email confirmation
- Booking management (view/cancel)
- Stable end-to-end workflow

9.b Backlog Features

- Real airline API integration
- Payment gateway (Stripe/PayPal)
- User profile editing
- Admin dashboard for flight and seat management
- Loyalty program with reward points
- Round-trip and multi-city bookings
- AI-driven price prediction engine

Prototypes for these features can be added to illustrate future expansion.

10. Lessons Learned / References

10.a What Went Well

- Team collaboration through Teams, GitHub, and DevOps
- Clear division of work across backend, frontend, and seat/booking module
- Successful implementation of dynamic pricing and ranking
- Clean UI with responsive search and booking flow

10.b Challenges and How They Were Overcome

- **Discount logic complexity:** refined through testing and formula adjustments
- **Seat map rendering:** solved with reusable JS components
- **Email confirmation errors:** addressed with proper SMTP configuration
- **Time constraints:** resolved by prioritizing core booking features

10.c What We Would Do Differently

- Start backend earlier to reduce integration pressure
- Adopt a frontend framework (React) for improved UI scalability
- Add automated unit tests for search and booking flows
- Plan deployment strategy earlier

10.d Software or Approaches Considered but Rejected

- Using a real airline API — rejected due to cost and complexity
- Using PostgreSQL or MySQL — SQLite was simpler and sufficient
- Implementing a full payment gateway — beyond project scope

10.e Anything Helpful for Future Development

- Maintain updated UML + ERD diagrams
- Add CI/CD pipelines
- Log all booking and user activity for analytics
- Introduce accessibility and ADA compliance checks

10.f References

List references such as:

- Node.js and Express.js documentation
- SQLite documentation
- Nodemailer documentation
- MDN Web Docs (HTML/CSS/JS)
- GitHub issues and StackOverflow community

11. Closing Comments

The **LastChance Air** project allowed the team to integrate system analysis, full-stack development, UML modeling, database design, and agile project management. It provided a realistic, hands-on experience in designing and implementing an information system from start to finish.

The IS 699 course successfully bridged theory and practice, enabling teamwork, structured documentation, and iterative system building. Future classes may benefit from earlier DevOps onboarding and more examples of end-to-end project frameworks.

12. Software Delivery

Repository URL: https://github.com/LokeshD28/lastchance-air.git

13. Application Delivery

The application currently runs locally. Future deployment to cloud platforms such as Azure, Render, Netlify, or AWS is feasible and recommended for broader accessibility.