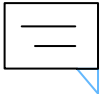


# Flight Booking Simulator With Dynamic Pricing



# Outline

1. Introduction
2. Overview
3. About Milestones
4. UI Demo
5. Use Cases
6. What new in project
7. Pros and Cons of new features
8. Challenges and solution
9. Future Scope
10. Conclusion



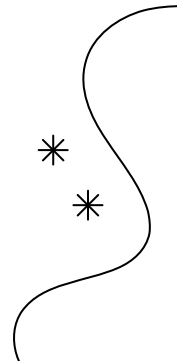
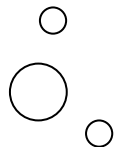
# Introduction

- A web-based **Flight Booking Simulator** that allows users to search and book flights.
- Implements **dynamic pricing** to simulate real-world fare variations.
- Built using **JavaScript**, and **FastAPI (Python)** for backend services.
- Integrated with **User Sign-In** for user authentication.
- Aims to deliver a **realistic and interactive flight booking experience**.



# Overview

- Simulates a real-world **flight booking system**.
- Integrates **dynamic pricing** that changes fares in real time.
- Prices vary based on **demand, seat availability, and seat type (like economy or business)**.
- Demonstrates how airlines use **data-driven strategies** to optimize revenue.
- Provides an interactive platform for **searching and booking flights**.





# About Milestones



## Milestone 1: Core Flight Search & Data Management (Weeks 1-2)

**Objective:** Build the backend foundation for flight data handling.

### Key Work Done:

- **Database Setup:** Designed schema (origin, destination, times, price, seats) using SQLite/PostgreSQL and added sample flight data.
- **REST APIs:** Developed with FastAPI/Django for retrieving and searching flights by origin, destination, and date; added sorting and input validation.
- **Simulated Airline Data:** Created mock API feeds to simulate real airline data.

### Output:

- Working flight database,
- functional flight search API
- simulated data sources.



## Milestone 2: Dynamic Pricing & Demand Simulation(Week 3 - 4)

**Objective:** Add real-time pricing behavior to make the simulator dynamic.

### Key Work Done:

- **Dynamic Pricing Logic:** Prices change based on remaining seats, time to departure, demand, and fare tiers.
- **API Integration:** Integrated pricing engine with the flight search API for live fare updates.
- **Background Simulation:** Simulates demand and seat availability over time.
- **Optional Fare History:** Tracks historical price changes for analytics.

### Output:

- Dynamic pricing API.
- demand simulator.
- optional fare history tracker.



### Milestone 3: Booking Workflow & Transaction Management (Weeks 5-6)


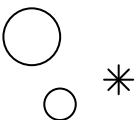
**Objective:** Implement a complete booking process with concurrency safety and confirmation tracking.

#### Key Work Done:

- **Schema Design:** Created booking schema with flight ID, passenger, seat, status, and price.
- **Booking Flow:** Added steps for flight selection, passenger info, and simulated payment.
- **PNR Generation:** Generated unique PNR after successful booking.
- **Concurrency Control:** Ensured seat consistency using DB transactions or locks.
- **APIs:** Built booking cancellation and history retrieval endpoints.

#### Output:

- End-to-end booking workflow.
- Concurrency-safe seat reservation.
- PNR assignment and booking storage.
- Functional cancellation and history retrieval.



## Milestone 4: User Interface & API Integration (Weeks 7-8)

**Objective:** Develop frontend and connect backend APIs for a seamless user experience.

### Key Work Done:

- **UI Design:** Built responsive frontend using HTML/CSS/JS or React.
- **API Integration:** Connected flight search and booking APIs.
- **Dynamic Pricing:** Displayed live fare updates in search results.
- **Booking Flow UI:** Designed stepwise booking with PNR confirmation.
- **Receipt Generation:** Enabled PDF/JSON booking receipts.
- **Testing:** Performed integration and final UI polish.

### Output:

- Functional frontend integrated with backend.
- Real-time pricing and booking confirmation.
- Downloadable receipts.
- Complete, testable user experience.





# Use Cases

## 1. Search Flights

Users can find available flights by selecting route, travel date, and class.

## 2. Select Seats

Passengers choose their preferred seats, and prices update instantly.

## 3. Book Flight

System collects traveler details and confirms the booking.

## 4. Dynamic Pricing

Ticket fares change in real time based on demand and seat availability.

## 5. Payment Simulation

A mock payment system generates a valid booking with a PNR number.

## 6. Booking History & Cancellation

Users can view previous bookings or cancel them easily.





# What's new in project





# Pros and Cons of new features

## Advantages of Implemented Features

- Dynamic Pricing
- REST APIs Integration
- Database-Driven System
- Real-Time Search & Filtering
- Simulated Airline Feeds
- Booking Workflow
- Frontend Integration

## Disadvantages of Implemented Features

- Higher System Complexity
- Longer Development & Testing Time
- Increased Maintenance Effort
- Risk of Performance Slowdown
- Data Dependence
- User Learning Curve
- Potential Scope Creep





# Challenges and Solution

① **Challenge:** Handling real-time flight data and availability

**Solution:** Used optimized APIs to manage live flight and seat updates.

② **Challenge:** Designing a dynamic and fair pricing system

**Solution:** Implemented demand-based fare adjustment using seat occupancy logic.

③ **Challenge:** Preventing overbooking and data conflicts

**Solution:** Applied strict booking validation and atomic seat updates.

④ **Challenge:** Achieving fast and scalable backend performance

**Solution:** Built using FastAPI for high-speed, asynchronous processing.

⑤ **Challenge:** Ensuring data accuracy and consistency

**Solution:** Used Pydantic models for validation and structured data handling.





# Future Scope

## Google-Sign

Integrate Google sign-in authentication for users.

## Integration with Real-Time Flight Data APIs

Connect simulator with live airline or airport APIs to display real flight schedule

## User Personalization and Recommendation

Use user history and preferences to recommend best flight options, destinations.

## Multi-Currency Support

Allow users to pay in different currencies.

## Enhanced Security and Authentication

Incorporate OAuth 2.0, two-factor authentication to improve user trust and protect sensitive information





# Conclusion

