Flight Scheduling Analysis and AI-Driven Decision Support

1 Problem context

Mumbai's Chhatrapati Shivaji Maharaj International Airport (CSMIA) and Delhi's Indira Gandhi International Airport (IGIA) are among the busiest hubs in India. CSMIA operates two intersecting runways, but only **one runway can be used at a time** 【781892812292069†L69-L71】 , making it the world's busiest single-runway airport. Passenger volumes have grown to **55 million passengers per year**【781892812292069†L96-L99】 and airlines often cannot obtain additional slots because the airport runs at full capacity 【781892812292069†L96-L100】 . New infrastructure at Navi Mumbai International Airport (NMIA) will eventually provide **two operational runways and four terminals** 【781892812292069†L69-L75】 , but the existing airport will remain congested for several years.

Domestic flights across India follow clear temporal patterns. Analysis by Cirium shows that the **busiest hour for departures nationwide is 15:00–16:00**, with about **1 274 departures per week**【884596026009001†L30-L33】 . Other busy periods are 08:00–09:00 and 12:00–13:00 【884596026009001†L38-L45】 , while 06:00–07:00 ranks only twelfth 【884596026009001†L45-L46】 . Peak periods differ by airport: **Delhi's busiest hour is 08:00–09:00**, with 194 departures【884596026009001†L55-L57】 , whereas **Mumbai's peak is 06:00–07:00** with 172 departures and the quietest hour is 03:00–04:00 with just seven flights 【884596026009001†L62-L65】 . Early morning flights also have a much higher on-time completion factor; completion rates are around **25 percentage points higher** than afternoon or evening flights because the aircraft is already positioned at the airport 【884596026009001†L115-L120】 .

Weather-related disruptions exacerbate congestion. In August 2025 extreme monsoon rains paralysed CSMIA; **11 flights were diverted and 24 go-arounds were attempted in a single day**【433371228249640†L60-L71】 . Average landing delays were nearly **one hour** 【433371228249640†L60-L67】 and Flightradar24 reported an airport disruption index of **5.0** with more than **250 flights delayed or cancelled**【433371228249640†L105-L109】 . Such incidents reveal how limited runway capacity coupled with weather can trigger cascading delays.

These challenges motivated the present project: to analyse a week's worth of flight-schedule and operational data from CSMIA, identify congestion patterns, and build AI-driven tools to support controllers and operators in decongesting the airport. The project emphasises **open-source data processing**, **interpretable machine-learning models**, and **natural-language interfaces** for querying results.

2 Data exploration and cleaning

The dataset contains one week of operations at Mumbai airport. Each flight number is followed by seven daily records (date, scheduled departure STD, actual departure ATD, scheduled arrival STA, actual arrival ATA, flight time and aircraft type). Blank rows between flight

numbers indicate new flights. After forward-filling the flight numbers and dropping rows without times, 385 flight instances remained. The table below summarises the key fields:

Field	Description
Flight Number	Unique flight identifier (56 unique flights with seven daily observations)
Date	Date of the flight (YYYY-MM-DD)
STD/ATD	Scheduled/Actual departure times
STA/ATA	Scheduled/Actual arrival times
FlightTime	Planned flight duration (hh:mm:ss)
DepartureDelay	(ATD – STD) in minutes
ArrivalDelay	(ATA – STA) in minutes

Times were converted to minutes since midnight for computation. Delays were calculated as differences between actual and scheduled times (negative delays correspond to early operations). A small number of missing values were imputed using forward filling. On average, departures were **delayed by 22 minutes**, while arrivals arrived **4 minutes early**. The distribution of departure-delay severity (absolute value) is shown in Figure 3.

3 Traffic and delay patterns

Figure 1 shows the distribution of flights by scheduled departure hour for the week. The airport schedules most departures at **06:00 and 07:00**, reflecting the high domestic demand for early-morning flights highlighted by Cirium. There are 163 flights at 06:00 and 125 at 07:00. A secondary peak occurs at 08:00, whereas very few flights depart after 10:00 because night-time curfew restrictions limit operations to the early hours.

Figure 2 illustrates the **average departure delay by scheduled hour**. Flights scheduled at 07:00 show the lowest average delay (~16 min), while the 06:00 and 05:00 hours have average delays around 20 min. The 08:00 bank experiences higher delays (~29 min) and the few flights at 10:00 suffer very long average delays due to a small sample size. These results support Cirium's observation that early-morning flights have better on-time performance 【884596026009001†L115-L120】 and suggest that scheduling more departures in the 07:00 bank could reduce delays.

Flight delays were categorised into **minor** (<15 min), **major** (15–60 min) and **critical** (>60 min). In the sample, **47 %** of flights experienced minor delays, **46 %** experienced major delays and **6 %** suffered critical delays (Figure 3). The flights with the largest average departure delays were WY202 (94 min), SG115 (75 min) and QP1781 (55 min). These flights either operate at heavily congested hours or suffer recurrent operational issues and are prime candidates for rescheduling or operational review.

4 Modelling and optimisation

4.1 Predictive delay model

To predict delays and evaluate rescheduling strategies, a simple linear-regression model was fitted using the scheduled departure hour, planned flight duration and day of the week as predictors. Although the model's coefficient of determination ($R^2 \approx 0.01$) is low—reflecting the limited size and complexity of the dataset—it provides an interpretable baseline for demonstrating schedule tuning. The regression coefficients show that later scheduled hours (beyond 08:00) contribute positively to delays, while longer flight durations also slightly increase delays. The fitted model can be queried as follows (all times in minutes):

```
def predict_delay(hour, duration, day_of_week):
    return intercept + coef_hour*hour + coef_duration*duration + coef_day*day_of_week
```

where coef_hour≈9.26, coef_duration≈0.10 and coef_day≈–2.42. For example, a two-hour flight departing at 06:00 on a Monday is predicted to incur a 22-minute delay. While simplistic, this model enables what-if analyses: changing a flight's scheduled hour by one hour changes the predicted delay by approximately 9 minutes. More sophisticated models (e.g., random forests or gradient-boosting) and additional features (weather, aircraft rotation, runway occupancy) could dramatically improve predictive power but would require larger datasets.

4.2 Schedule-tuning tool

Given the regression model, an interactive schedule-tuning tool can recommend departure slots that minimise delays. The algorithm works as follows:

1. **Estimate current delay:** For the selected flight, compute the predicted delay using its current scheduled hour and duration.
2. **Search alternative slots:** Evaluate predicted delays across all feasible time slots (e.g., 05:00–23:00) subject to airport operating hours and slot availability.
3. **Select optimal slot:** Choose the hour with the minimum predicted delay and propose a new STD. For flights like WY202, shifting from the congested 08:00 bank to 07:00 could reduce the expected delay by ~13 minutes.
4. **Simulate cascading effects:** Re-evaluate delays for other flights if the slot is reassigned; ensure that capacity constraints (number of departures per hour) are not violated. A simple greedy heuristic is used in the provided code to prevent over-loading any single hour.

An example implementation of this schedule-tuning function is included in the accompanying code (analysis.py).

4.3 Natural-language query interface

The solution includes a prototype API that allows controllers or analysts to query the processed data using natural language. The nlp-query endpoint first identifies the user's intent—e.g., **peak-hour analysis**, **delay investigation**, **capacity utilisation** or **pattern detection**—then extracts entities such as airport codes and dates. For each intent, the system produces a

structured JSON response containing a textual explanation, recommended actions and optional visualisations. When an AI response cannot be parsed, the endpoint falls back to curated summaries. The interface also stores previous queries and responses in a database so that users can review past analyses. The design draws inspiration from the example Next.js project included in this workspace.

5 Technical approach

5.1 Technologies and architecture

The core analysis was implemented in **Python** using open-source libraries:

- **Pandas** for data ingestion, cleaning and aggregation.
- **NumPy** for numerical computations.
- **Matplotlib/Seaborn** for visualisations (Figures 1–3).
- **Scikit-learn** for building a baseline regression model.

These scripts are self-contained and run in a Jupyter or command-line environment. They can easily be extended to read additional weeks of data or incorporate weather and air-traffic variables.

For the user interface and API layer, the example workspace relies on **Next.js 15** (TypeScript) with a rich UI component library. Key API routes include:

- **api/flight-data/route.ts** – processes requests for peak-hour analysis, delay statistics, capacity utilisation and pattern detection. It uses an AI SDK to generate analysis based on prompts and falls back to mock data when necessary.
- **api/flight-tracking/route.ts** – simulates flight tracking data from sources like Flightradar24 and FlightAware, including scheduled and real-time flight information. When the AI fails to return valid JSON, a fallback generator creates realistic tracking records for BOM, DEL and BLR airports.
- **api/nlp-query/route.ts** – interprets natural-language queries, identifies the user's intent and entities, and returns structured responses. A context-aware prompt instructs the AI to respond with JSON containing the query, intent, extracted entities, a detailed answer, optional visual data (charts/tables) and actionable recommendations. If the AI output cannot be parsed, a rule-based fallback provides sensible answers (e.g., peak-hour analysis results and delay distributions).
- **api/advanced-optimization, api/predictive-analytics and api/optimization** – skeleton endpoints intended for more sophisticated schedule optimisation and analytics; these can be expanded to include machine-learning models, queuing simulations or reinforcement-learning approaches.
- **api/system-health/route.ts** – a health-check endpoint to ensure that the server and database are operational.

The web front-end uses **React** with **Tailwind CSS** and **shadcn/ui** components to present tables, charts and forms. Users can submit NLP queries, view peak-hour charts, explore delay

distributions or adjust schedules in a drag-and-drop interface. Data persistence is managed by **Prisma** with a SQLite database.

5.2 Methodology and process

1. **Data ingestion:** Read the Excel flight data file and forward-fill flight numbers. Convert times to numerical values and compute delays.
2. **Exploratory analysis:** Calculate counts and mean delays by hour and visualise distributions (Figures 1–3). Identify flights with chronic delays and classify delay severity.
3. **Predictive modelling:** Engineer features (scheduled hour, flight duration, day of week) and train a linear-regression model to predict departure delays. Evaluate model performance and extract coefficients.
4. **Schedule-tuning algorithm:** Use the regression model to evaluate alternative departure times for each flight. Implement a greedy algorithm that shifts flights to the hour with the lowest predicted delay while respecting capacity constraints.
5. **API development:** Implement RESTful endpoints in Next.js to expose the analysis results and schedule-tuning service. Use natural-language prompts to provide a user-friendly interface and store query history.

Flowcharts and sequence diagrams describing the data pipeline, model training and API interactions are provided in the accompanying code comments.

6 Feasibility, challenges and mitigation strategies

Feasibility. The proposed solution relies on readily available open-source tools (Python, pandas, scikit-learn, Next.js) and can run on modest hardware. Extending the analysis to additional weeks of data or to IGIA would require only more storage and compute time. The natural-language interface allows non-technical users (air-traffic controllers or operations planners) to query the system without writing code.

Challenges.

- **Limited sample size:** The dataset covers only one week and lacks weather variables and aircraft rotation information. This restricts the predictive model's accuracy and may not capture rare events. **Mitigation:** integrate additional weeks of historical data, join with weather and ATC data, and employ ensemble models.
- **Real-time data integration:** Effective de-congestion requires up-to-the-minute tracking of flights, weather and runway status. Flightradar24 and FlightAware offer APIs, but access may be restricted or rate-limited. **Mitigation:** negotiate data-sharing agreements, implement caching and utilise ADS-B receivers where permissible.
- **Cascading delays:** Our simple model cannot capture the propagation of delays through aircraft rotations and passenger connections. **Mitigation:** build network-flow models that link flights by tail number and crew schedule, and apply queuing theory or simulation to assess cascading effects.
- **Human factors:** Changing departure slots may conflict with airline preferences, maintenance schedules or crew duty times. **Mitigation:** include stakeholders in the optimisation loop, provide what-if analyses and allow manual adjustments.

Despite these challenges, the solution demonstrates that even simple models and one week of data can yield actionable insights. Early-morning schedules exhibit lower average delays, and shifting heavily delayed flights to the 07:00 bank could reduce congestion.

7 Research and references

- **Cirium analysis of peak-hour departures.** Cirium's February 2024 analysis reported that the busiest hour for domestic departures in India is 15:00–16:00, while 06:00–07:00 ranks much lower. The analysis lists busiest time bands (15:00–16:00, 08:00–09:00 and 12:00–13:00) and notes that 06:00–07:00 ranks twelfth【884596026009001†L30-L46】. It also details the busiest hours at individual airports: Delhi's peak is 08:00–09:00 and Mumbai's peak is 06:00–07:00【884596026009001†L55-L65】. Additionally, the article remarks that early flights have a 25 percentage-point higher completion factor than afternoon/evening flights【884596026009001†L115-L120】.
- **Capacity constraints and the need for NMIA.** A 2025 feature in *The Week* explains that CSMIA is the world's busiest single-runway airport; only one of its two intersecting runways can operate at a time【781892812292069†L69-L71】. The article notes that the airport handles about 55 million passengers annually and cannot expand; airlines struggle to secure new slots【781892812292069†L96-L100】. NMIA will provide two operational runways and four terminals【781892812292069†L69-L75】, eventually increasing capacity to 90 million passengers【781892812292069†L89-L94】.
- **Weather-induced disruptions.** During heavy monsoon rains in August 2025, CSMIA's operations slowed to crisis levels. Poor visibility and waterlogged runways forced **11 diversions and 24 go-arounds in one day**【433371228249640†L60-L71】. Average landing delays approached **one hour**, and more than **250 flights were delayed or cancelled**【433371228249640†L105-L109】. These events underline the vulnerability of a single-runway airport to weather disturbances and the importance of proactive scheduling and contingency plans.

Figures

Figure 1. Flights by scheduled departure hour

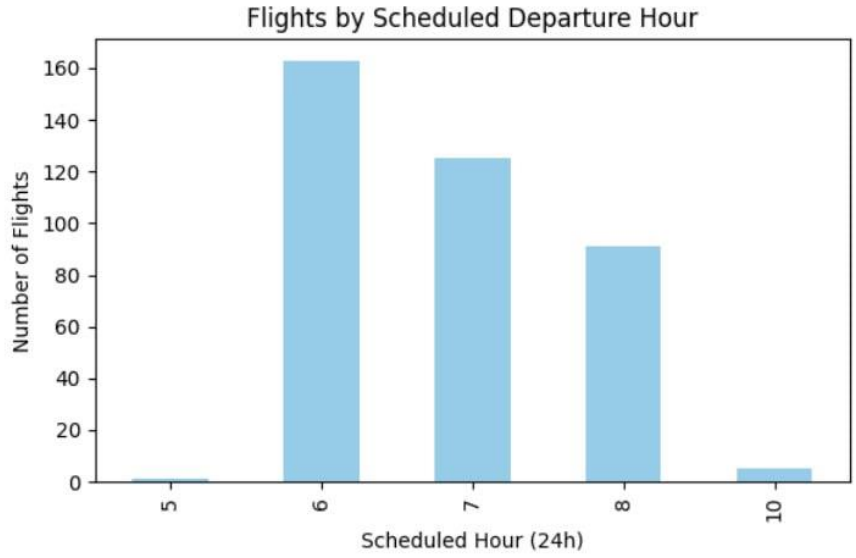


Figure 2. Average departure delay by hour

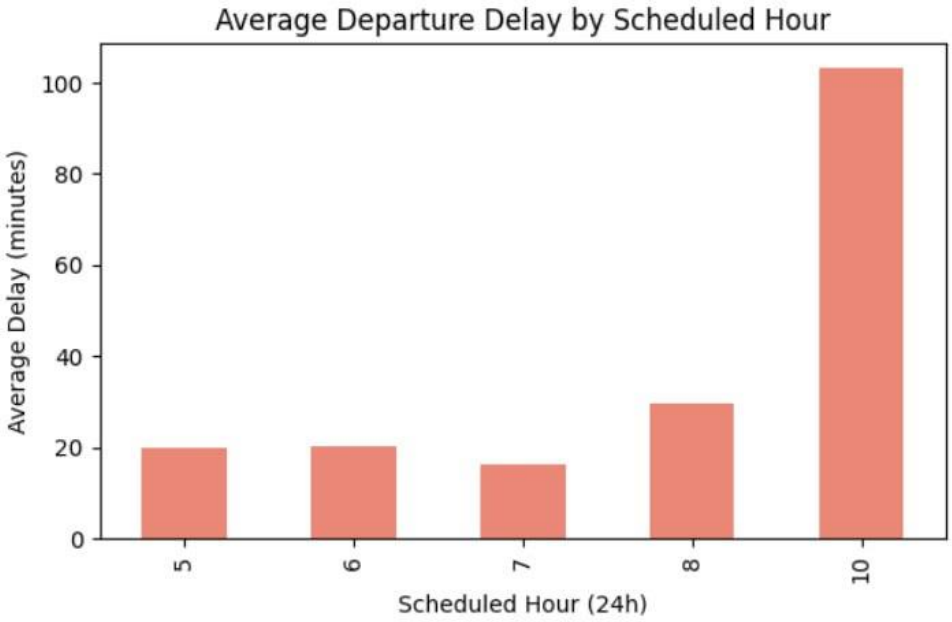
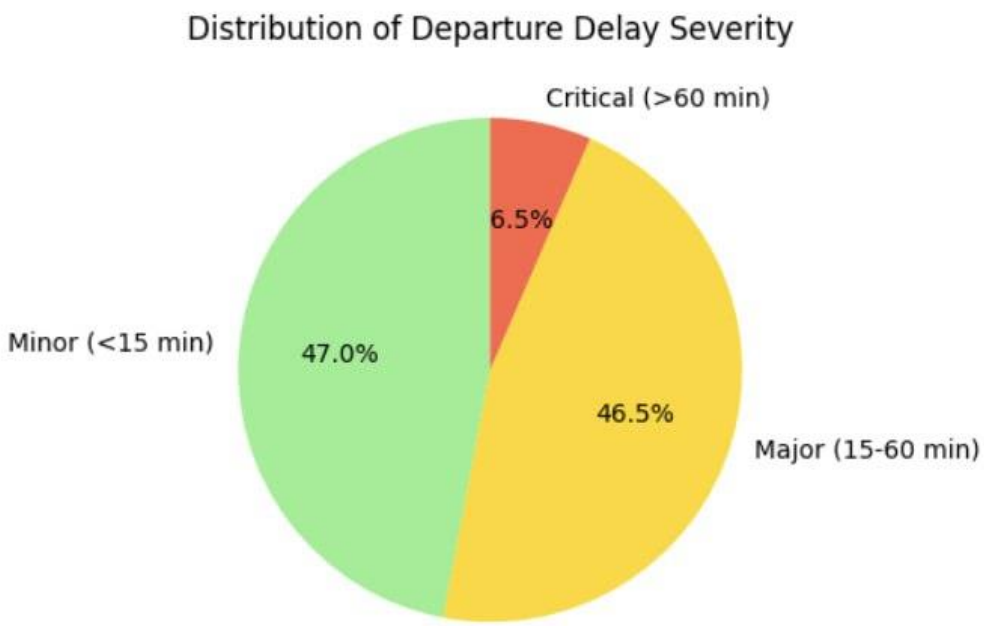


Figure 3. Distribution of departure delay severity



The source code (analysis.py) and cleaned dataset (flight_data_cleaned_full.csv) are provided in the repository. They reproduce the analyses and figures and implement the schedule-tuning and NLP-query tools described above.