Solution Design Problem

Introduction

Your task is to design a system to implement a Flight Information Display system at an Airport - you know, one of those information boards that show when flights are departing. The goal isn't to have expert domain knowledge (so no points for researching other solutions), but to propose a solution to the problem as described and have discussions around tradeoffs. You can assume any technology stack or implementation that you're comfortable with, just note them down and explain why.

While there is no enforced time limit, we don't expect you to spend more than an evening of your time working on this problem.Just let us know how long you spent on it, so we can take that into account. There are more requirements than you can probably solve for in a reasonable time, so break up the problem appropriately - solving well for a clearly defined MVP is preferred over a solution that solves everything poorly.

Problem Statement

The task is to design a system that meets the following requirements:

1. For this exercise, we'll consider ONLY departing flights (not arrivals as well)
2. There is a flight schedule, which defines when the regularly scheduled flights occur - for example, "Air New Zealand has a flight NZ0128 that flies to Melbourne (MEL) at 6:30am on Monday, Wednesday and Friday"
3. The airlines keep the schedule up to date when they make schedule changes.
4. The flight display has a list of upcoming departures.
5. Each flight has the following properties
    a. An Airline
    b. Flight Number
    c. Destination
    d. Scheduled Departure Time
    e. Estimated Departure Time
    f. Actual Departure Time
    g. Flight Status, which is one of:
        i. On Time
        ii. Check In
        iii. Boarding
        iv. Departed
        v. Cancelled
        vi. Delayed
    h. Departure Gate (assigned once the flight enters "Boarding" status)
6. The big ticker board in the airports will get the information from your system over a web API.
7. The flight information needs to be viewable over the internet (so people can check their flight status before coming to the airport)
8. The internet accessible view of flight information must deal with very large traffic spikes for when a storm or other event means lots of people check flight status.
9. Passengers can subscribe to a particular flight and receive push notifications when it's status or details change/
10. Airlines must not be able to update the flight information for other airlines. 11. The interface to update the flight information must not be accessible to the internet.
11. The interface to update the flight information must not be accessible to the internet.

Deliverables

1. A design for how the data should be modeled.
2. A design for how the system would be broken down into components and what each of those components would do, along with technology choice.
3. A description of which requirements are met and not met, any trade-offs considered and any assumptions made.
4. A rough estimate for how long it would take you to implement the system described.

Some tips to help make your submission great

- The software that you'll be delivering should be thought of as a product, so don't just provide a solution that works, provide a solution that will continue working for a long lifespan.
- We don't expect all requirements to be met, but be specific about which requirements are met, and which are not.
- Be clear about your assumptions, not only the assumptions that we should make while reading your submission, but also what assumptions you made while coming up with the solution.
- Consider how your solution will be tested and whether that testing could be automated.
- Consider how your solution will be deployed. Can this be automated and to what degree.
- Break down your estimates and include the size of the team you'd think you'd need. We're not specifically looking for a nice, round figure in terms of time.
- Consider your delivery methodology of choice. How will we deliver this solution.
- Ensure you provide the reasoning behind your technology choices.
- Last but definitely not least, think about the user. They are the reason why we make software after all.

HAVE FUN!