TicketOverflow

Software Architecture

April 29, 2025

Evan Hughes & Richard Thomas

1 Brief

Recent controversy has brought attention to the monopoly held by certain ticketing companies. In hopes of an eventual collapse of the current monopoly, we aim to develop TicketOverflow, an online concert ticket booking system.

Task You are working for TicketOverflow, a new competitor in the online ticket booking space. TicketOverflow uses a microservices based architecture to implement their platform. The CEO saw on your resume that you took Software Architecture and has assigned you to design and implement the ticket purchasing service. This service must be scalable to cope with a large influx of bookings.

You also need to implement a mock of the users service. This service is being implemented by another team at TicketOverflow. You need to implement a mock to allow your ticket purchasing service to be tested. Your mock of this service will use a hard-coded list of users.

Requirements As you may be aware, ticket booking platforms can have intense peaks of traffic. At the scale that TicketOverflow hopes to operate, it would be inappropriate to run the servers required to meet demand at all times. Thus, our service must be elastic — able to scale up to meet demand and able to scale down to preserve costs.

2 Outline

Introduction (5 minutes)

Introduction to the brief and resources, including the API specification[1], hamilton ticket generation constraints[2], and quality scenarios in section 3.

Planning (10 minutes)

In small groups, discuss the following issues or any others you think are relevant to designing the service.

1. What are the key requirements introduced by the quality scenarios?
2. What strategies can you implement to support these scenarios?
3. What AWS resources would prove helpful?
4. What are the likely bottlenecks?

[1] https://csse6400.uqcloud.net/api/ticketoverflow
[2] https://github.com/CSSE6400/hamilton

Design (20 minutes)

In your group, design an appropriate architecture for TicketOverflow. You need to consider the flow of an API request through your system, use the API specification, hamilton ticket generation constraints, and quality scenarios in section 3 to ensure all use cases have been considered.

Presentation (15 minutes)

In the remaining time, each group should present their proposed architecture design. This is an opportunity for discussion amongst the class to point out limitations of the proposed system designs.

3 Quality Scenarios

Q1: Small concert As TicketOverflow is starting out, it will need to scale to the size of a small concert. Playhouse at QPAC is one of the first venues to sign up for the service. Their customers purchase tickets in advance over a long period of time (i.e. several weeks), but with higher demand closer to the concert date.

Q2: Pre-sale for Hamilton Hamilton is now showing at the Lyric Theatre at QPAC and the pre-sale tickets are released for a limited time. Tickets are in high demand, and there are a large number of customers trying to purchase tickets at the same time.

Q3: General sale for Hamilton The general sale for Hamilton has started, and there is a surge of customers trying to purchase tickets at the same time. This leads to a high volume of traffic on the website, which puts a strain on TicketOverflow.

Q4: Seating plan launch The frontend team at TicketOverflow have developed a new ticket purchasing interface that shows users the seating plan before they purchase. A large concert is being used to trial this new feature. You need to ensure that the rendering of the seating plan does not interfere with the high volume of associated ticket purchases.

Q5: Evening shows at QPAC All QPAC venues are now supported in TicketOverflow. Each evening there are multiple concerts that run concurrently. TicketOverflow must be able to handle a steady stream of parallel purchases, ticket generation, and seating plan generation.

Q6: Priority tickets Tickets recently went on sale for a large concert in four months time. Some of the ticket purchasers are downloading their tickets in advance. Meanwhile an evening show for a smaller concert is starting in a few minutes. The evening show attendees should be able to download their tickets without being impacted by the later concert.

Q7: Copyright infringement Due to a copyright notice, "Elsa on Ice", has to be renamed to "Bob on Ice". Unfortunately, this is a last minute name change while many users are logging in to generate their tickets. Your system must ensure that they are not shown incorrect tickets and gracefully handles re-generation of tickets.

You are not required to queue up re-printing all the tickets that have already been printed. As indicated in the API specification for the update concert endpoint (`/concerts/id PUT`[3]), "any existing tickets that have been generated need to be removed." This means that, for the concert which is updated, all tickets that have a `PRINTED` status must have that status changed to `NOT_PRINTED`. You need to ensure that tickets

[3] https://csse6400.uqcloud.net/api/ticketoverflow/#api-Concert-be

with a `PENDING` status cannot print tickets with outdated information. It is the client's responsibility to determine that the ticket status has changed and to re-print the ticket. Your system has to manage the load.

Q8: Taylor Swift tour Taylor Swift has chosen TicketOverflow for the release of her *Next* Eras Tour. It is estimated that there will be about 2.8 million purchases during this time. Your service should do its best to remain responsive during the launch of the tour.