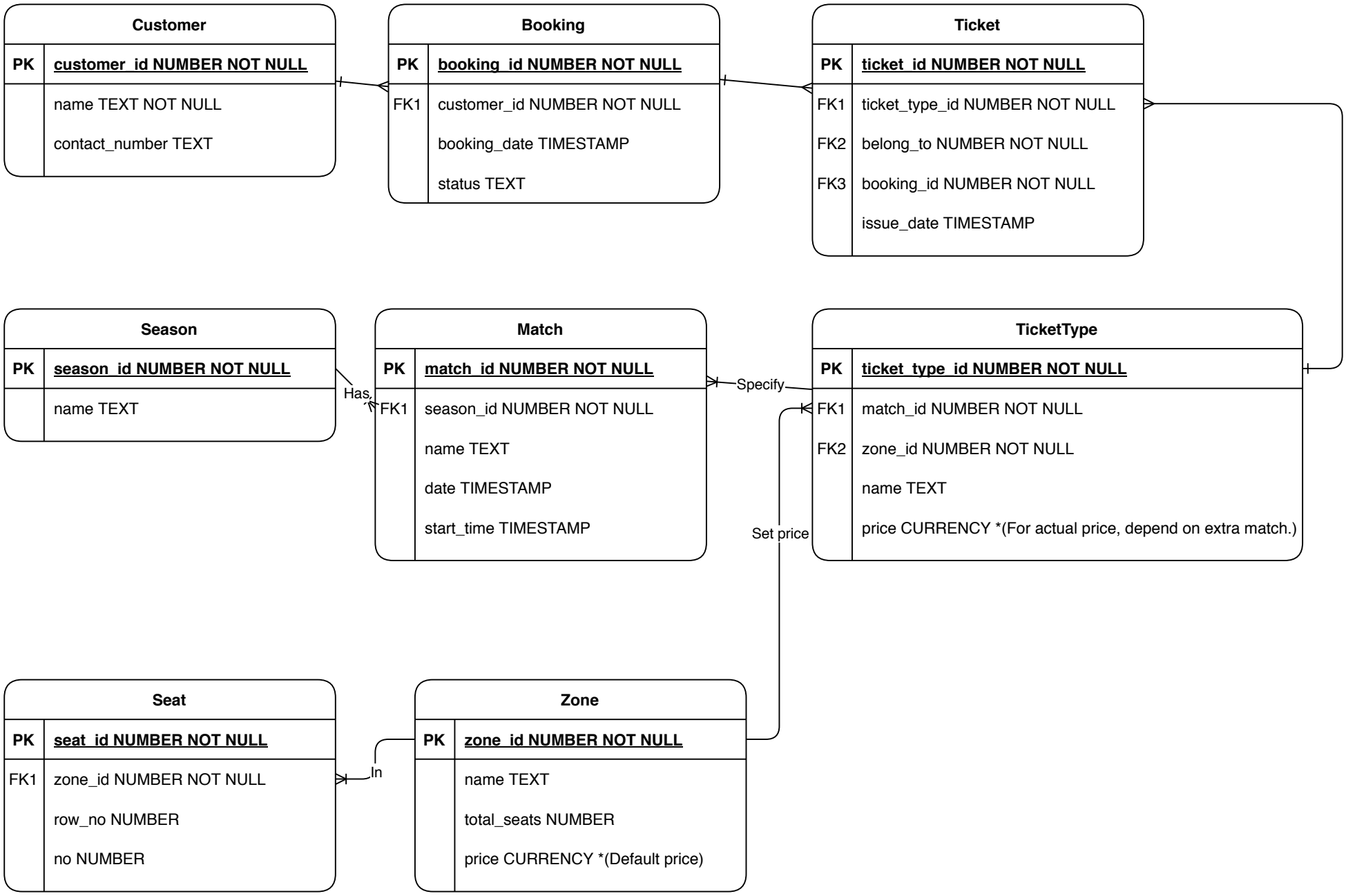
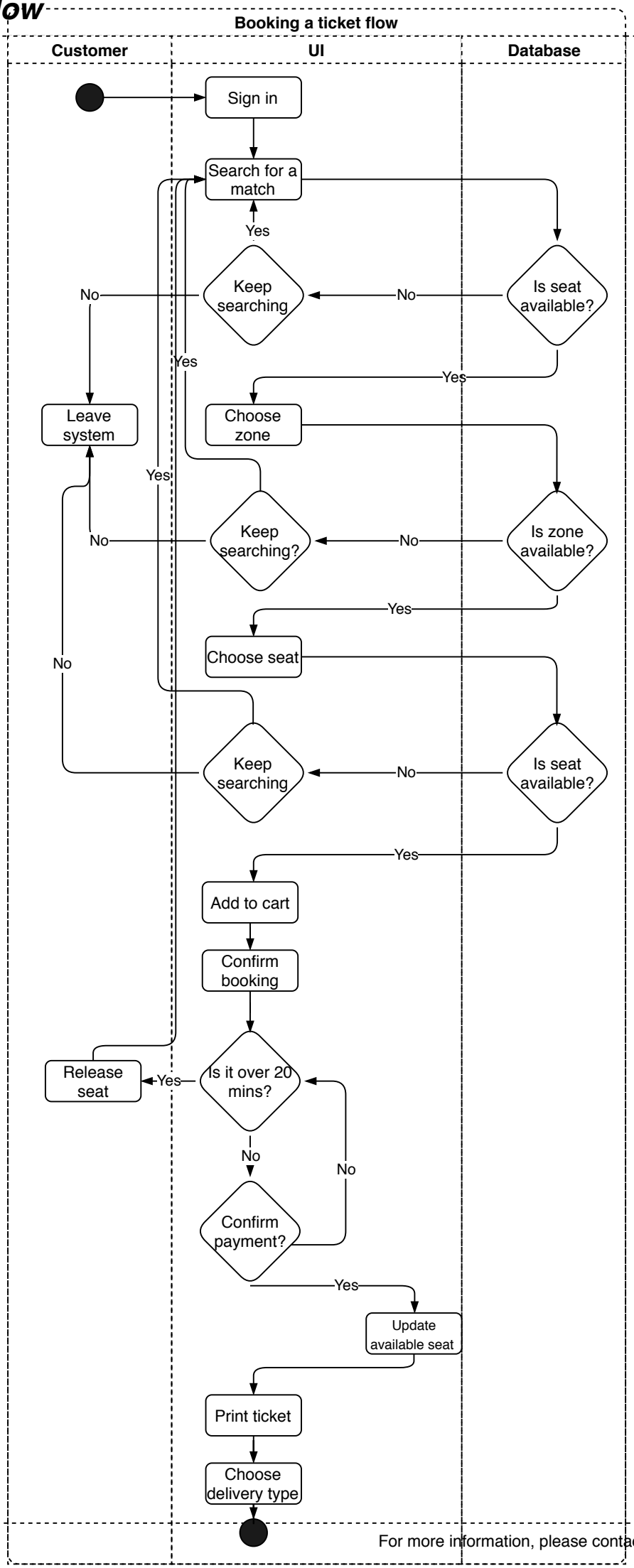
Database Design



For more information, please contact cray.j.thiemsert@gmail.com

Ticket Booking Flow



Problem

Crowed of football fan

1) Customer จำนวนมากเข้าใช้งานพร้อมๆ กัน จนบางทีทำให้ระบบล่มได้ ส่วนใหญ่จะเกิดเมื่อมีการเปิดให้จองแมตซ์ที่คนให้ความสนใจเป็นจำนวนมาก

Solution

การออกแบบระบบการเข้าคิวอย่างมีประสิทธิภาพเพื่อลด traffic ของ server ในเวลานั้นๆ โดยที่พยายามหลีกเลี่ยงและกระจายปริมาณของ customer ที่จะเข้ามาใช้ resource เดียวกันของระบบ ยกตัวอย่างเช่น

1) เมื่อ customer กดเลือกแมตซ์ที่ต้องการแล้วให้ระบบบันทึกเวลาที่กดเลือกแมตซ์นั้นแล้วส่งข้อมูลนี้ไปให้ background service เพื่อทำการคำนวนคิวตามเวลา แล้วส่ง customer ไปที่หน้า "รอคิว"

2) ที่หน้ารอคิวจะเป็นหน้าง่ายๆ ที่แจ้งคิวของ customer คนนั้น ว่าเหลืออีกกี่คิวถึงจะเป็นคิวของ customer คนนั้นที่จะได้ทำการเลือกจองตั๋ว

3) เมื่อถึงคิวแล้วจะมีการแจ้งเตือนให้ customer ว่าเค้าสามารถเข้าระบบไปเพื่อทำการจองตั๋วได้ โดยระบบจะกำหนดระยะเวลาให้เพียงพอแก่การทำการจองตั๋วคนละ 1 transaction เท่านั้น

4) ถ้า customer ไม่ได้เข้าระบบมาทำการจองตั๋วเมื่อถึงคิวของตน ในระยะเวลาที่กำหนด ก็จะข้ามไปยัง customer ในคิวถัดไปขึ้นมา

Database & Cloud Service

เลือก Firebase หรือ AWS เพราะเป็น cloud platform ที่มีเครื่องมือและ library ให้ค่อนข้างเพียงพอต่อการพัฒนา web และ mobile application
โดยมีการให้บริการครบวงจรทั้ง database, storage, hosting, cloud function, authentication, analytic, machine learning เป็นต้น

ซึ่งเราสามารถประหยัดเรื่องการไม่ต้องดูและลงทุนทางด้าน server hardware เอง และ เรามั่นใจในเรื่องความเสถียรและความปลอดภัยเรื่อง data center ได้ในระดับหนึ่ง

และในช่วงแรกของการพัฒนาเราอาจไม่มีค่าใช้จ่ายในการใช้บริการ เนื่องจาก data usage ที่ใช้ยังอยู่ในสเกลที่ไม่มากจนถึงระดับที่ต้องเสียค่าบริการ การเสียค่าบริการนั้นจะเกี่ยวเนื่องกับปริมาณข้อมูลที่ใช้งานซึ่งเราสามารถเลือกขนาดที่เหมาะสมกับโปรเจ็คของเราได้ในภายหลัง

Communication Protocol

เราอาจใช้ประโยชน์จาก **WebSockets** ที่มีประสิทธิภาพในการทำงานแบบ 2 ways communication แบบ Real-time ที่ดีกว่า HTTP Request/Response

กระบวนการทำงานที่สำคัญ

เมื่อได้รับหัวข้อของระบบที่จะต้องทำการพัฒนา จะต้องเข้าใจอย่างถ่องแท้ถึง pain point และความต้องการของ user ทุกฝ่ายที่เกี่ยวข้องเพื่อจะนำข้อมูลนั้นไปใช้ในการออกแบบระบบเพื่อสร้างประสบการณ์การใช้งานที่น่าพึงพอใจแก่ user

ยกตัวอย่างระบบการจองตั๋วฟุตบอลโลกนี้ จะมี pain point หลักๆ คือการใช้งานอย่างหนาแน่น และการแย่งจองตั๋วจาก user จำนวนมากในเวลาไล่เลี่ยกัน จนทำเกิดปัญหาหน้าจอค้างเพราะระบบจัดการ traffic จำนวนมากไม่ทัน

ถ้าเราวิเคราะห์การใช้งานการจองตั๋วของ user ให้ดีก็จะเห็นว่าจุดที่ทำให้เกิดการส่ง request ซ้ำๆ วนไปและไม่ยอมจบ process ง่ายๆ คือการหาที่นั่งที่ว่างและอยู่ในบริเวณที่ต้องการ ปัญหาจะเกิดขึ้นเมื่อ user กดจองที่นั่งนั้นๆ ไปแล้ว แต่กลับปรากฏว่าที่นั่งนั้นไม่ว่างเนื่องจากถูก user อื่นจองไปแล้ว ก็ต้องกลับมาวนหาที่ว่างต่อไป

ซึ่งในกระบวนการนี้ถ้าเป็นการใช้ HTTP Request/Response ก็จะเกิดปริมาณการส่ง request/response มหาศาล จนเกินปริมาณที่ server รองรับได้

หนึ่งในแนวทางที่ควรนำมาใช้แก้ปัญหานี้คือทำอย่างไรให้ user ลดการเลือกที่นั่งผิดพลาดจากการถูกจองไปก่อนแล้วให้น้อยที่สุด โดยการนำเทคนิคของ WebSockets มาใช้ เพื่อส่งข้อมูลให้ client จำนวนมากแบบ real-time เมื่อเราลดการวนเลือกหาที่นั่งว่างได้ก็จะช่วยทำให้ user ใช้เวลาอยู่ในกระบวนการจองตั๋วน้อยลงและเสร็จกระบวนการได้เร็วขึ้น และจะช่วยทำให้ traffic ที่หนาแน่นของ user มีการไหลออกจากระบบได้ดีขึ้น

Scalability

ในส่วนของความสามารถในการรองรับการขยายตัว เพื่อรองรับปริมาณของธุรกรรมที่เพิ่มมากขึ้นนั้น WebSockets 1 IP Address สามารถรองรับได้ 65,536 sockets หรือก็คือ 65,536 clients ในเวลาเดียวกัน ถ้าเราจำเป็นที่จะต้องขยายปริมาณที่รองรับเราก็แค่ใช้เทคนิคในการทำ load balance เพื่อเพิ่มจำนวน IP Address ขึ้นมาเช่น Elastic Load Balance ของ AWS หรือ Software Load Balancers เจ้าอื่นๆ ก็ได้