Making Live Happen: Software Engineering at Ticketmaster

Kitty Shi
College Park Scholars - Science and Global Change
Computer Science (Data Science Track) and Biological Science
kyshi874@terpmail.umd.edu
CPSG230
Scholars Academic Showcase: May 9, 2025

Introduction

During the summer of 2023, I interned as a software engineer at Ticketmaster on the Archtics Integration team. Under the guidance of our mentor, Talha Mahammad, I learned and applied full stack development knowledge to simulate a ticketing platform. There I also designed an AWS service system with Nithika Ramanathan, a fellow intern studying at UMD, to improve the management of errors in the existing infrastructures.

Internship Site

The Archtics Integration team's mission is to create and deploy APIs essential to the data-driven solutions offered to Ticketmaster clients in the live entertainment industry.

Contact Information:
Ticketmaster Reston Office
11921 Freedom Dr
Suite 800
Reston, VA 20190
Website: https://www.ticketmaster.com
Supervisors: John Kemprowski and Talha Mahammad

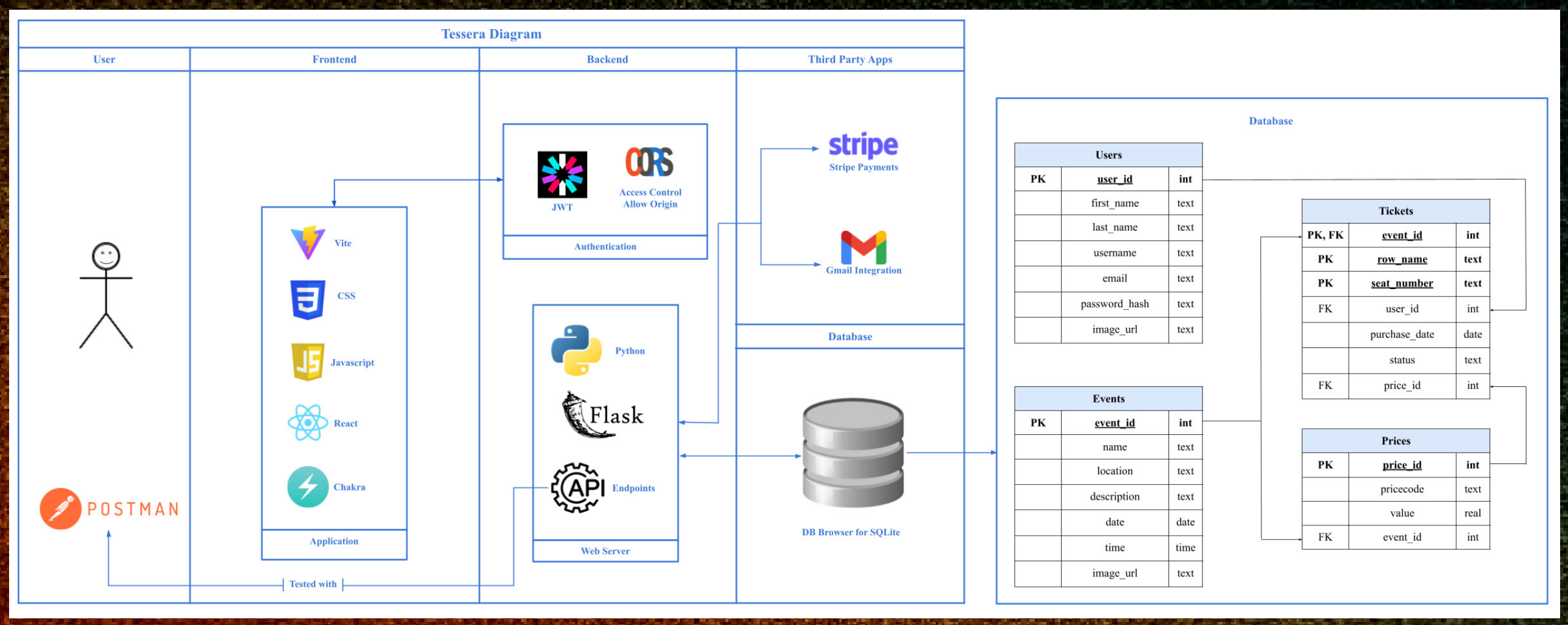
Full Stack Development

I created a ticketing platform, Tessera, with the base functions of account creation, login security, filtering, and ticket purchase essential to the service using the React framework with Javascript.

Database:
- Relational database with normalized data for data integrity
- Iterations of schema to improve efficiency and scalability

Backend:
- REST API using JWT, data extraction from cookies, and password hashing for security
- Database APIs to maintain data integrity as tickets are purchases Third party APIs used for payment and email confirmation of purchases (Stripe and Gmail Integration)
- SQL query efficiency and format of JSON

Frontend:
- Component organization to reduced redundant API calls
- Passing props, implementing callbacks and React hooks



Issue Confronting Site

Ticketmaster teams utilizes AWS services to manage responses for ticket purchases, surveys, and other applications. Erroneous messages temporarily stored in AWS Simple Queue Service (SQS) that fail to be validated by AWS Lambdas to be stored in databases end up in AWS Dead-letter Queue (DLQ).

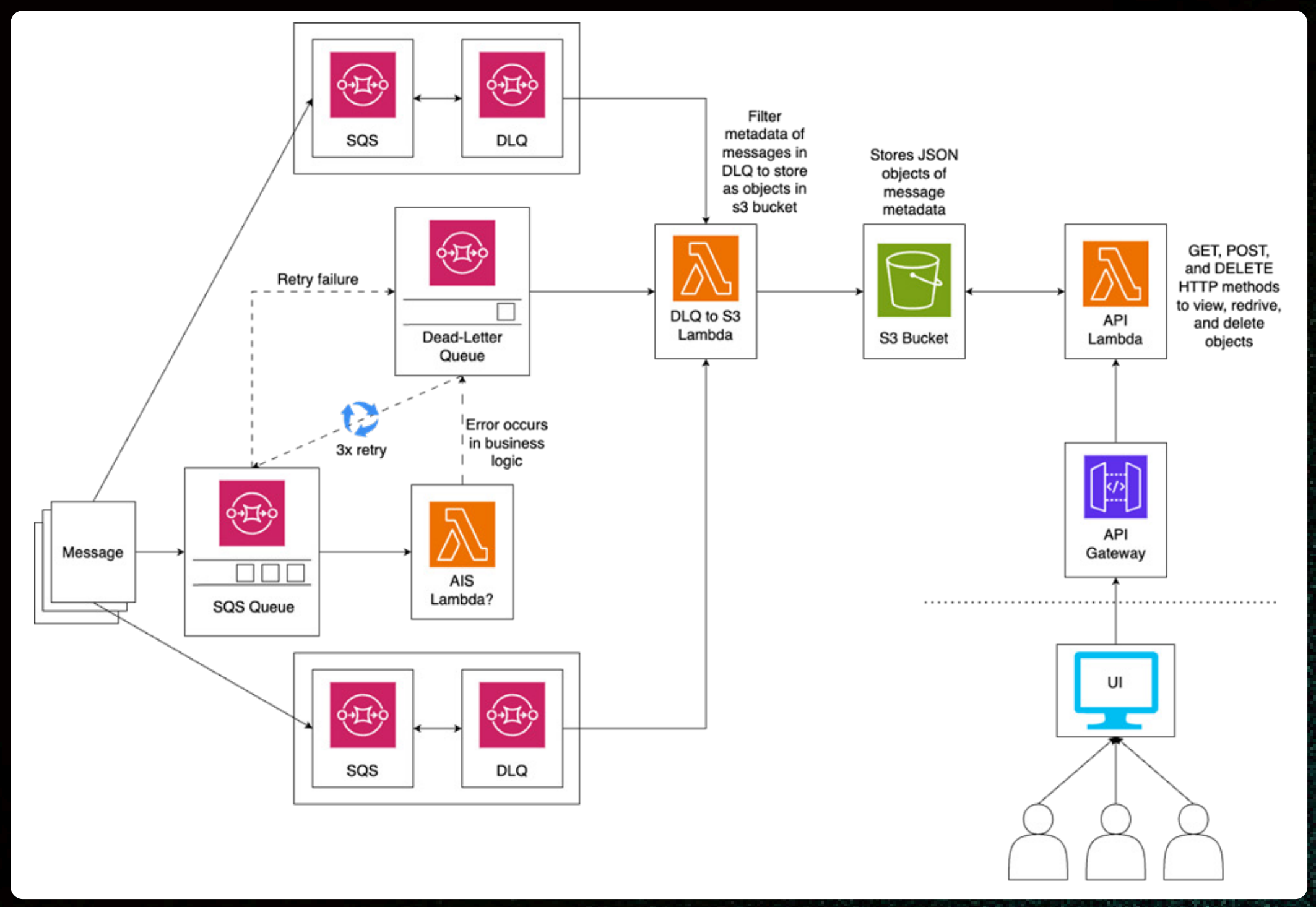
Since engineers must manually check DLQs there is an unknown amount of buildup of messages that are difficult to view, delete, and redrive without careful examination. The retention of messages in DLQ also only has a maximum of 14 days. There needed to be a more efficient system in place to manage erraneous messages

Activities

AWS SQS DLQ Metadata Managing System

We created a centralized system with Python using AWS Lambda, SQS, and Simple Storage Service (S3) to manage important metadata from messages collected in DLQs for ease of access of viewing and editing.

- Connected AWS resources through triggers
- Filtered DLQ message metadata to S3 JSON object in Lambda
- Designed APIs that displayed objects from S3 buckets


SQS
DLQ
Filter metadata of messages in DLQ to store as objects in s3 bucket
Stores JSON objects of message metadata
Retry failure
Dead-Letter Queue
DLQ to S3 Lambda
S3 Bucket
API Lambda
GET, POST, and DELETE HTTP methods to view, redrive, and delete objects
3x retry
Error occurs in business logic
Message
SQS Queue
AIS Lambda?
API Gateway
UI
SQS
DLQ

- Created a mock UI interface that engineers would use to interact with the objects
- Researched implemention of user authentication to limit user access to S3 buckets by authorization

Impact

While the managing system is incomplete due inefficiency of API calls, the connectivity design of the system is base for the engineers on the team to continue developing upon.

Through the development of Tessera I learned to code in JS, debug React, integrate systems through APIs. These skills can be transferred to any future works

Future Work

Adjustment to API calls to retrieve data from S3 can hopefully improve in time efficiency to where it won't time out on a bucket containing over 100,000 objects.

In my own time I would like to continue implementing features such as personalizd event suggestions to users based on their past purchases.

Acknowledgments

I want to thank John Kemprowki and the Archtics Integration Team at Ticketmaster for supporting and encouraging my growth as a software engineer during this unforgetable internship. Special thank you to Talha Muhammad for creating and curating lesson plans detailing the full stack development process and for your patience in answering every question regarding software engineering.
Additionally, thank you to Dr. Holtz and Dr. Merck for the guidance and lessons on the intersectionality of science, climate change, and sociopolitical events over the past two years

SCIENCE AND GLOBAL CHANGE