Ticketing App

Objective

> Users can list a ticket for an event (concert, sports) for sale

> Other users can purchase this ticket

> Any user can list tickets for sale and purchase tickets

> When a user attempts to purchase a ticket, the ticket is 'locked' for 10 minutes. The user has 10 minutes to enter their payment info.

> While locked, no other user can purchase the ticket. After 15 minutes, the ticket should 'unlock'

> Ticket prices can be edited if they are not locked

Reference : https://www.stubhub.com/

UI and UX screens

https://www.draw.io

GitTix   Sign Up   Sign In

Tickets For Sale

Basketball Game - $20

Classical Concert - $100

Rock Concert - $100

Football Game - $40

https://www.draw.io

GitTix   Sign Up   Sign In

Sign Up

Email

Password

Submit

https://www.draw.io

GitTix   My Orders Sell Tickets Sign Out

Tickets For Sale

Basketball Game - $20

Classical Concert - $100

Rock Concert - $100

Football Game - $40

ticketing.dev/tickets/J2342jpkg

GitTix   My Orders Sell Tickets Sign Out

Rock Concert

Price - $100
Status - Available

Purchase

/orders/alskdjfaksdjfkp

GitTix   My Orders Sell Tickets Sign Out

Purchasing Rock Concert

You have 30 seconds left to order

Pay

Page 1

https://www.draw.io

GitTix   My Orders Sell Tickets Sign Out

Purchasing F

You have 30 secon

Credit Card

Expiration

Submit

Pay

Page 1

/tickets/new

GitTix   My Orders Sell Tickets Sign Out

Create a Ticket

Title

Price

Submit

Resources

User

Name	Type
email	string
password	string

Ticket

Name	Type
title	string
price	number
userId	Ref to User
orderId	Ref to Order

Order

Name	Type
userId	Ref to User
status	Created | Cancelled | AwaitingPayment | Completed
ticketId	Ref to Ticket
expiresAt	Date

Charge

Name	Type
orderId	Ref to Order
status	Created | Failed | Completed
amount	number
stripeId	string
stripeRefundId	string

Micro Services

auth	Everything related to user signup/signin/signout
tickets	Ticket creation/editing. Knows whether a ticket can be updated
orders	Order creation/editing
expiration	Watches for orders to be created, cancels them after 15 minutes
payments	Handles credit card payments. Cancels orders if payments fails, completes if payment succeeds
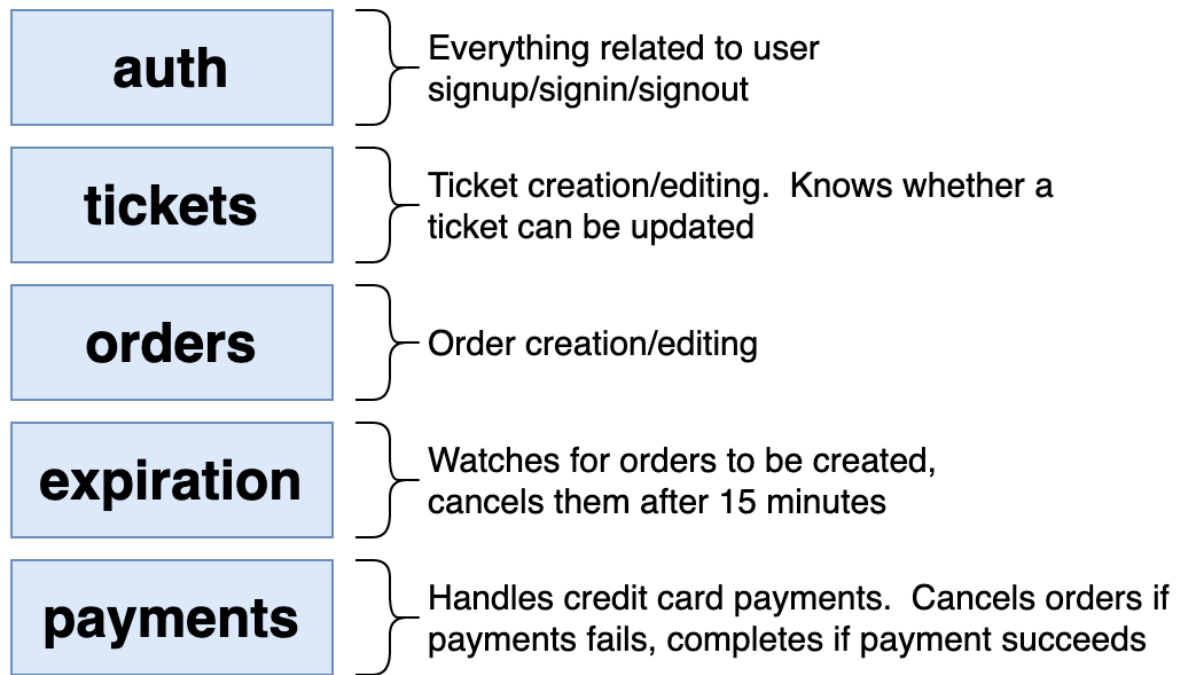
Flow diagram
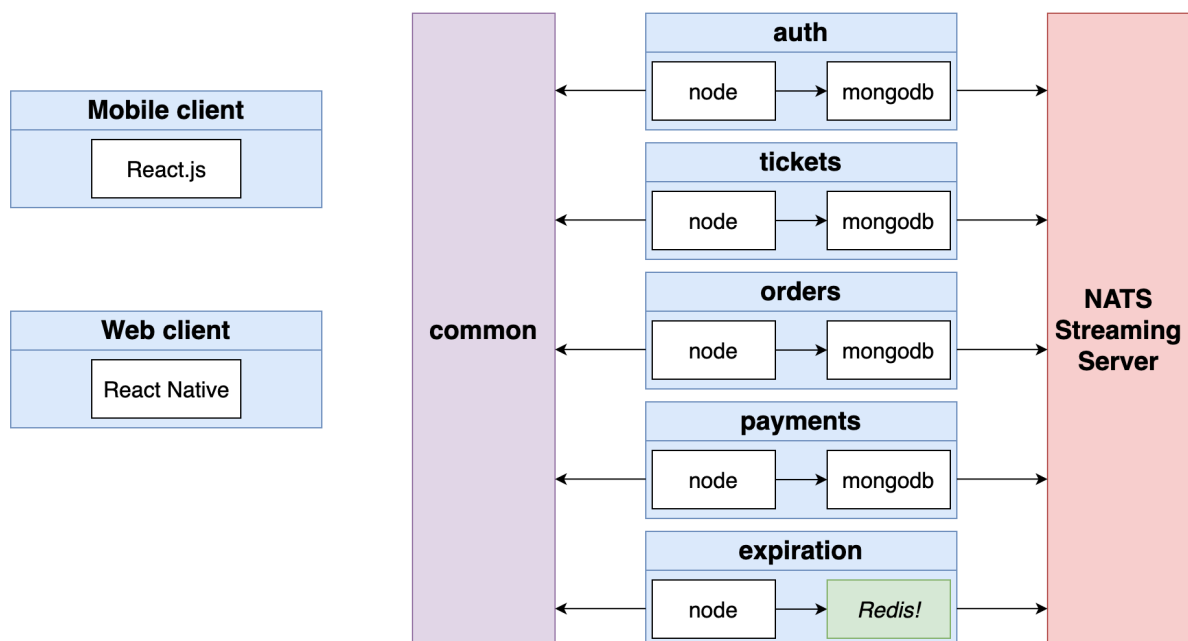
Application Events

UserCreated	UserUpdated

OrderCreated	OrderCancelled	OrderExpired

TicketCreated	TicketUpdated

ChargeCreated

a. Auth Service

| auth			
Route	Method	Body	Purpose
/api/users/signup	POST	{ email: string, password: string }	Sign up for an account
/api/users/signin	POST	{ email: string, password: string }	Sign in to an existing account
/api/users/signout	POST	{}	Sign out
/api/users/currentuser	GET	-	Return info about the user

Auth Service

Middlewares	Route Handlers	Models
requireAuth	signup	User
currentUser	signin	
	signout	
	current-user	

MongoDB for Auth Service

---
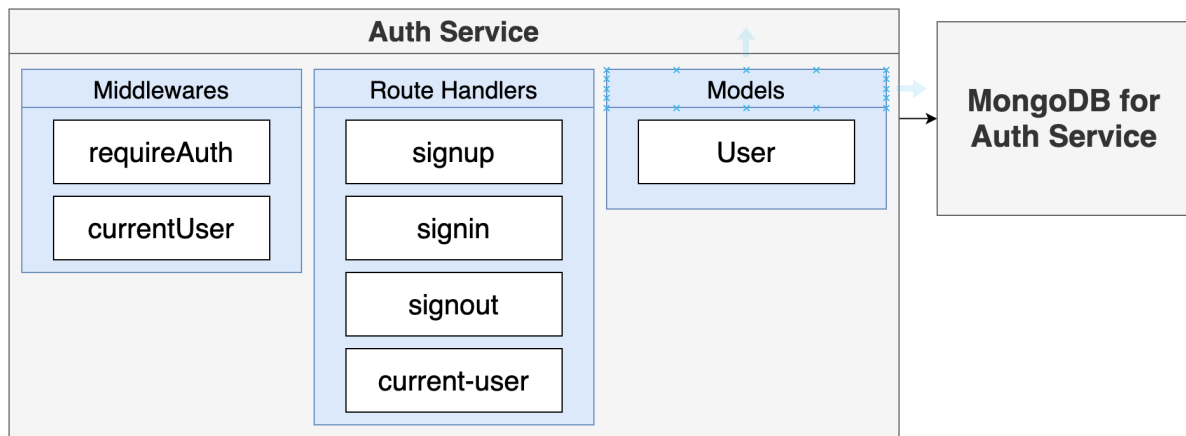
b. Ticketing Service

Tickets Service			
Route	Method	Body	Goal
/api/tickets	GET	-	Retrieve all tickets
/api/tickets/:id	GET	-	Retrieve ticket with specific ID
/api/tickets	POST	{ title: string, price: string }	Create a ticket
/api/tickets/:id	PUT	{ title: string, price: string }	Update a ticket

---

c. Order Service

---

orders			
Route	Method	Body	Purpose
/api/orders	GET	-	Retrieve all active orders for the given user making the request
/api/orders/:id	GET	-	Get details about a specific order
/api/orders	POST	{ ticketId: string }	Create an order to purchase the specified ticket
/api/orders/:id	DELETE	-	Cancel the order

d. Expiration Service
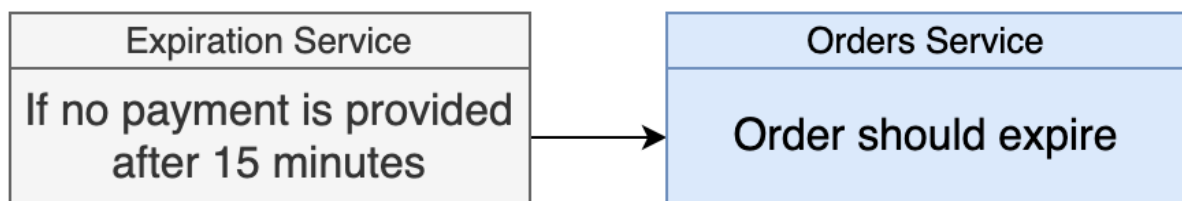
Things to consider and implement in this Ticketing App

- Keep complete code on github repository with branches
- Implement CI & CI pipeline
- Build as docker images and deploy on k8s platform using Local cluster