**Ticketing Website**

**INDEX**

1. Introduction
2. Technologies,Frameworks and Languages
3. Project Design
4. Resources
5. Services
6. Events
7. Structure
8. Auth Service

**Introduction**

Ticketing Website is going to be an ecomerce site, the main focus of this site is to see and buy tickets. User will be given a proper buying experience with respect to the seller details, users will be able to login with there credentials and can pay using credit card or other payment methods.

**Features**

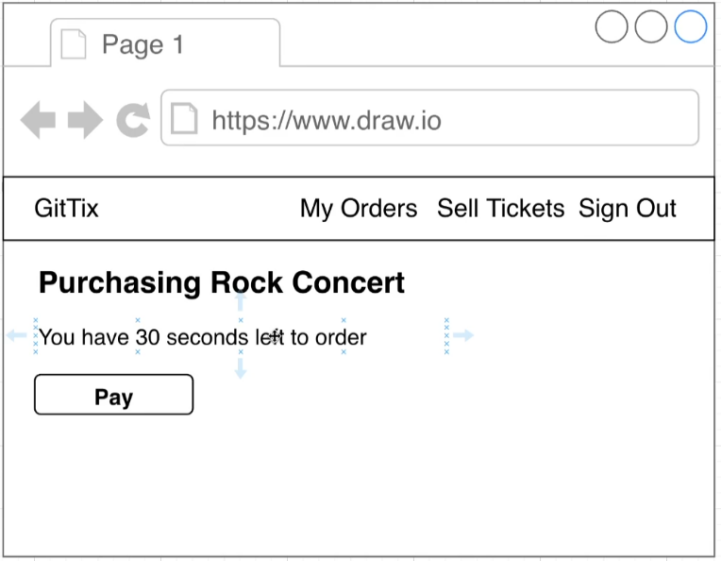
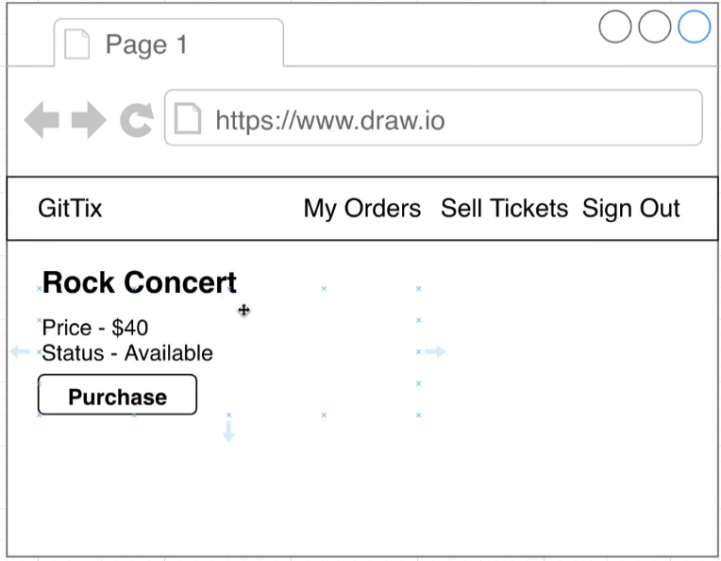
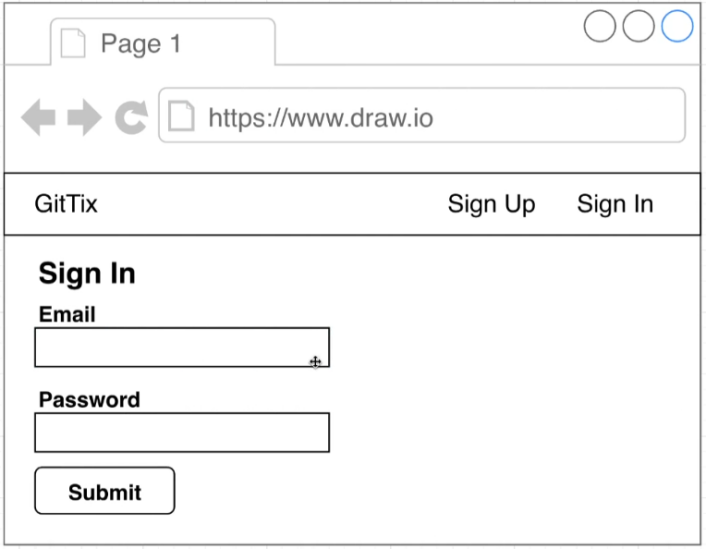
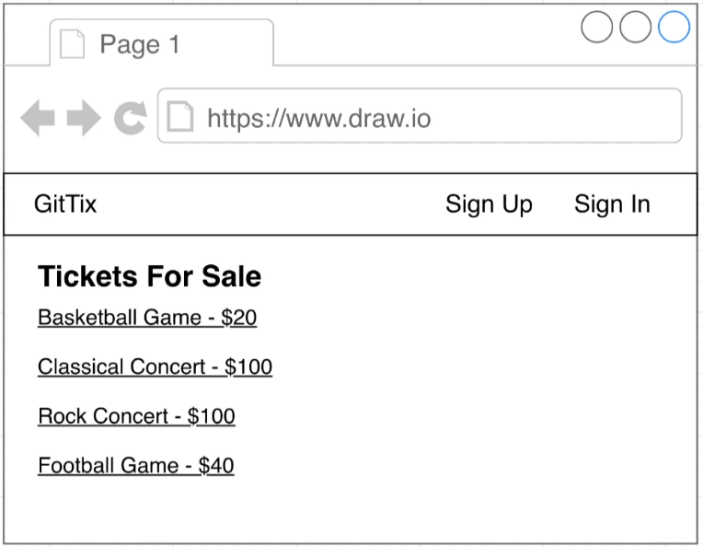
* Users can list a ticket for an event (concert, sport) for sale
* Other Users can Purchase these tickets
* Any user can list tickets for sale and purchase tickets
* When a use Attempt for purchase a ticket, the ticket is ‘**locked**’ for **15 minutes**. The user has 15 minutes to enter their payment 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 ‘l**ocked**’

**Languages, Frameworks and Technologies**

* HTML (Language)
* Css (Language)
* JavaScript (Language)
* TypeScript (Language)
* React (FrameWork)
* Node (Runtime Environment)
* Express (FrameWork)
* Mongodb (DataBase)
* Redis (Server)
* Kubernetes (Scaller and Maintainer)
* Docker (Scaller and Maintainer)
* Scaffold (Automator)
* minikube (VM for Kubernetes in Linux)

**Project Design**

Overview Steps



To Buy

**SignUp**, **SignIn**

After **SignIn**

Click **Purchase**

Click **Pay**

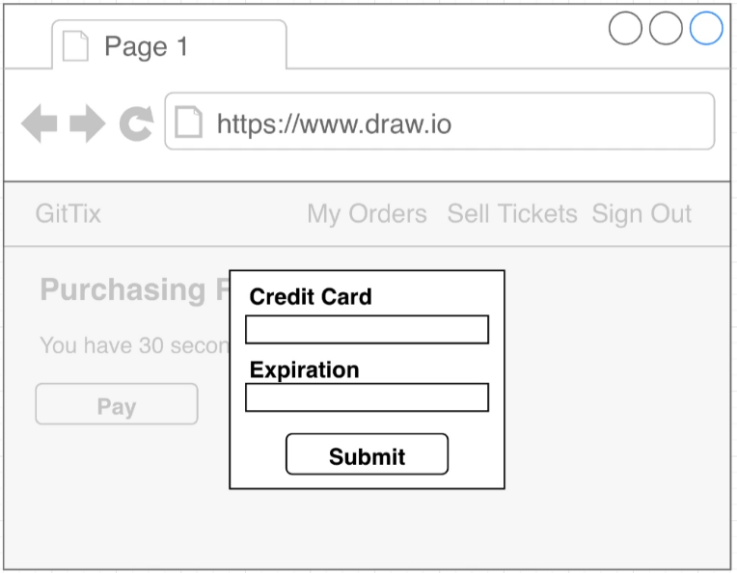
**For**

**Payment**

Enter

Payment

**Details**

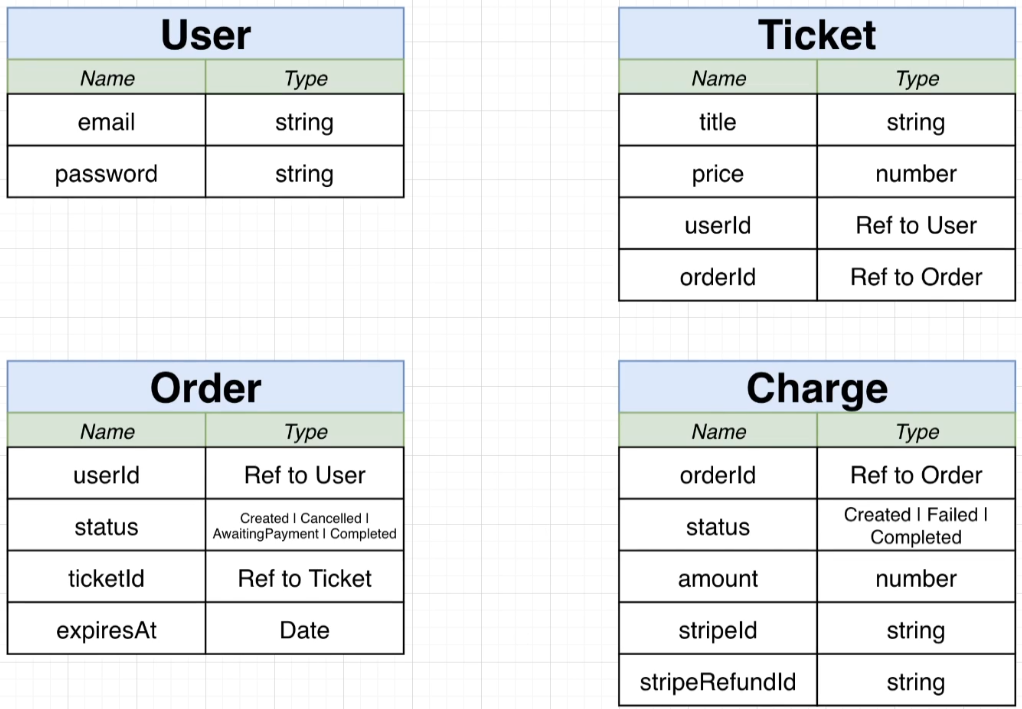


**Resources**

We are going to have different types of database structure which which can be or cannot be dependent on each other (Not necessory)

Following Tables or DataModels will be used in Future

**Tables**



**Services**

**Service** is an way through which frontend can interact with backend using restFull API’s

We are going to use many services,

It is not compelsory to create seperate independent server for each service, It depends upon the need for independency of each service

**In this project we will be using these services**

* **auth** -> Everything related to user signup/signin/signout
* **tickets** -> Ticket creation/editing. Knows weather 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 succeds

**Events**

Event is like an activity or a report which is used to communicate between services.

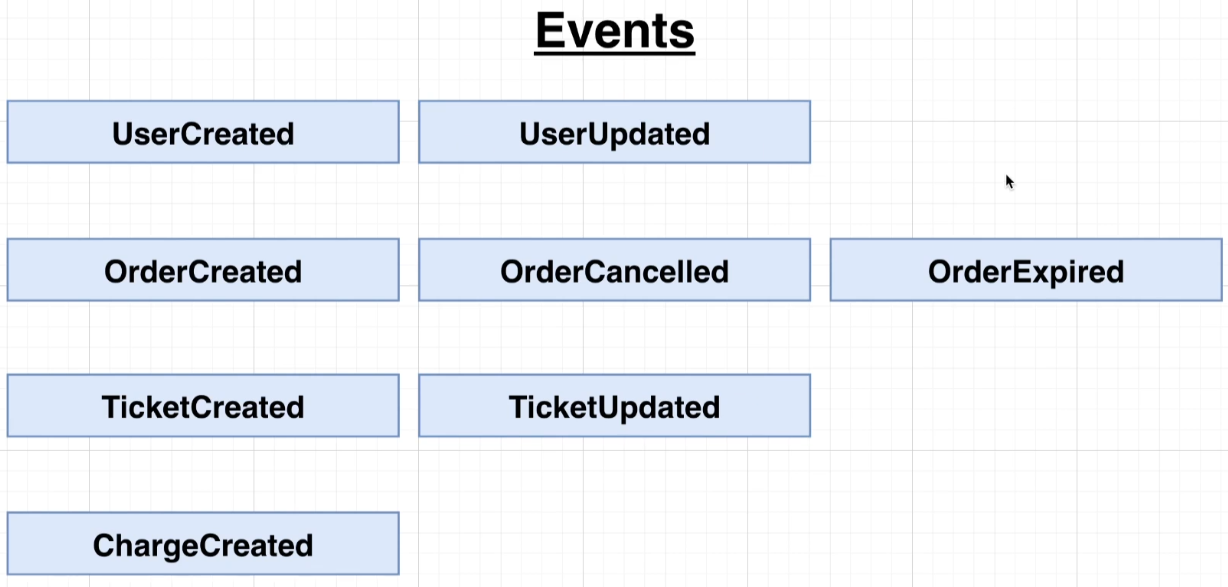
**1st type**

There can be some services which are event listner and will only perform if the event is used to triger these service untill then they can be used as restFull Api.

**2nd type**

Also There can be services which when become active will send event of its activness to every other services . So that if other service is dependent on this service then will start doing its work.

**For this Project We Will be having following events.**



**user service**

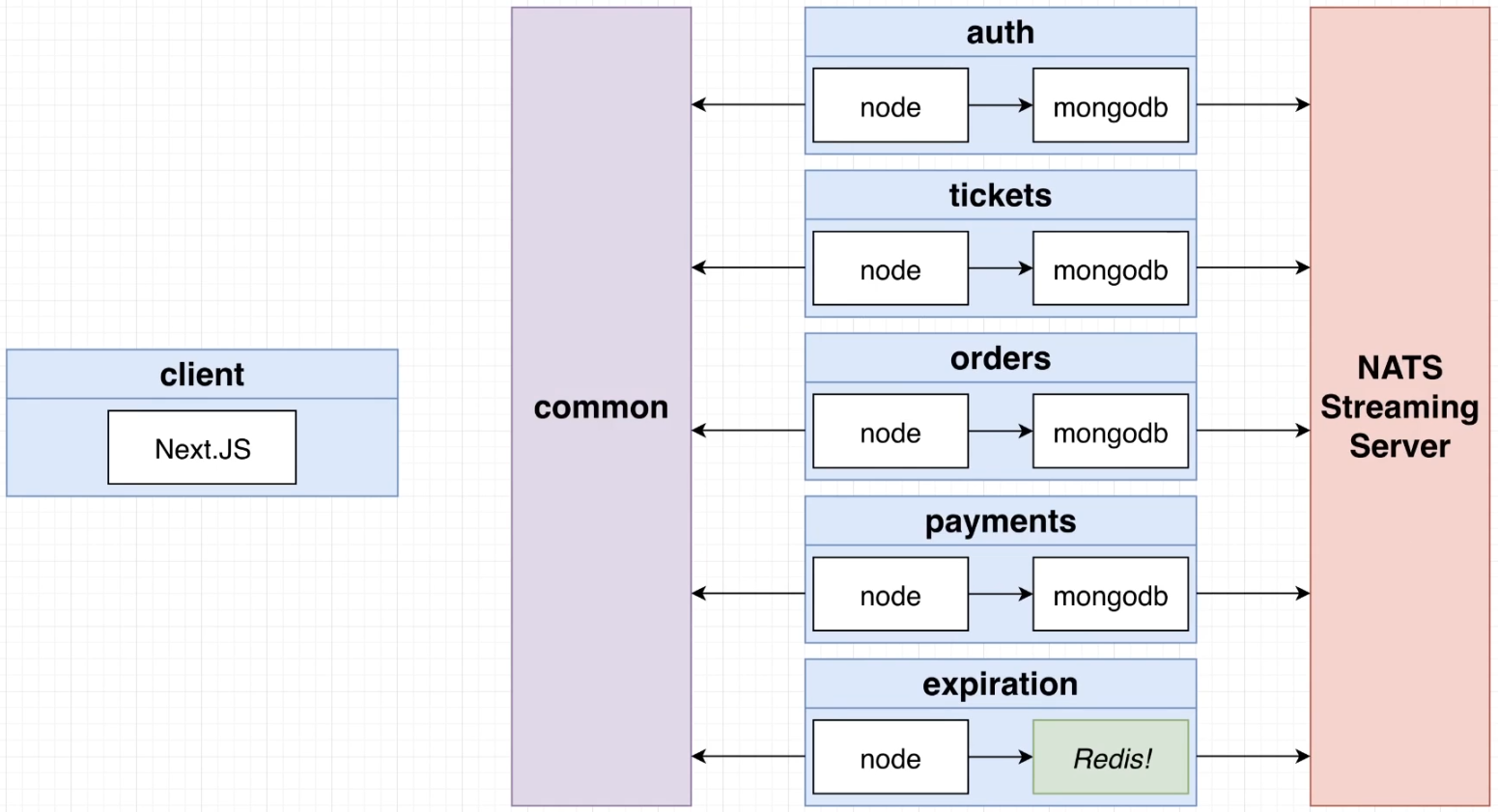
**order service**

**ticket service**

**charge service**

**Services**

**Project Languages and Service Structure**



**Client** -> Next.Js

**common** -> VM (for Event Communication (**Event Bus**))

**auth** -> node, mongodb

**tickets** -> node, mongodb

**orders** -> node, mongodb

**payments** -> node, mongodb

**expiration** -> node, Redis

**NATS Streaming Server**