Pricing Task Queue (PTQ)

- Operators to see and complete their tasks
  - By "market" → Phoenix / Sacramento (Opendoor region of operation)
- Tasks
  - Verify owner (how long eta?)
  - Verify bedrooms / bath etc
- Backend + Frontend

Opendoor Process
=> opendoor.com => seller flow => qualified => **make tasks in PTQ** => make a "final" offer => present to customer

Client side react app for seller (consumers from web)
Client side react app for operator (internal employees to opendoor0
Deployed to static site

Rest vs. GraphQL?

- Don't need GraphQL because not many relations

api.pricingtaskqueue.com => API Gateway => EC2

Deploy maybe using terraform or atlantis

TaskQueue (queue = in progress tasks)
    VerifyAttribute for (saleRequest, room, operator: null, completed: false, created_at)
    VerifyAttribute for (saleRequest, bathroom)
    VerifyAttribute for (saleRequest, owner)
Sql database
Tables (msql, RDS, postgres)
    House
        Rooms
        Bath
        Price
    SaleRequest
        belongToUser
        belongsToHouse
    BuyRequest
        belongTo user
    User
    Operator
    Task Table (update on firstCreation when sent to queue, when done at queue)
        taskId
        Market
        Operator
        Field: (sqfoot, rooms, bathroom, owner)
        Complete: true or false (inprogress = on queue but complete=false)

- Show me all unclaimed Phoenix tasks
  - Sort by deadline
  - Sort by created_at
- Claim a task (operator=null === unclaimed)

- Complete a task (task:complete)
- View completed tasks given a particular market (search through queue for all tasks with complete:false)

/task/:taskID
{
    "hey verify this owner is who they say they are",
    "status": "in progress | not started | completed"
}

Seller Client side React app sends requests with the data from seller to the backend

Backend has an endpoint called

(Rest api backend / flask)
post('/seller/submitSaleEstimate'):
    {Owner: 'sellerName',
    Bedrooms: 5
    Bath: 5}
    ….

Send to the taskQueue (SQS aws) //differetn queue for each market

Backend for operator frontend
    Reads from the task Queue

**Show me unclaimed tasks in Phoenix**
Operator Client(react app that reads from the taskQueue)
    UI that displays tasks.filter(byRegion and unclaimed)
    Read from Tasks Table for (complete:false, market: phoenix)

**Robin and Adnan both try to claim the same task at the same time in Phoenix**
First step is UI for adnan should update so button to claim is disabled
But if adnan sends click before that happens, then the endpoint to claim
When robin claims task, it sets the field operator to robin
    claimTask(operator, taskID)
        If (task.operator !== null) return 'error' //operator on client says "sorry already claimed"

claimTask
    => get task (this step actually locks the message)
    => check if task has operator: If (task.operator !== null)
    => task.claim(operator)

Client receaives N tasks and that size doesnt go up or down. Only status changes for each message

Pagination
- Prefetch 100 and store in redux / state management
- When i click the loadMore button on the 5th page of the first 100
- Look for the next 100 tasks after the first one by createdDate

**Operator finishes the task (task was verify owner, the seller was the owner of the property)**

VerifyTask(taskId, owner)
    Get Task(taskId)
    If (task,owner === owner)
        Mark Owner Table as complete and SAVE
        Remove task from queue
    Else
        addError to table or another queue to tell seller

=> od.com …… => PTQ => [ SQS / KAFKA ]
                => consume failed messages and email the owner
                => consumed successful sqfootage and notify our underwriters to begin the offer process

Previosu task:

- Verify sq_ft, owner said it was 100, it is actually 112

TaskTable
    Verified: false
    key: (sqfoot, rooms, bathroom, owner)
    Expected: 100 from seller
    actual: 112 from verifier

Two consumers

One checks for success
    Sends message to OD underwriter
One checks for failure
    Sends email to seller
    If (actual !== expected)
        Update the HouseTable with new value to prepopulate OD (models for pricing estimate)

Constraint:
    Tasks either live in database or database AND the queue
    If its on the queue, we can assume its in progress
    If its on the database only its either not started or done

Client polls backend (either by websockets, or http polling)
    This lets the backend push a message to the OperatorUI that says “reread QUEUE”
        Either tell the Ui to update Model or just tell UI to reread from backend
        This would refresh the list of tasks to say Claimed for the claimed ones

UI - [list of tasks [box says Claim Task]] onclick Claim,
*Operator claim a task
    Sends a request from client to operator-backend that marks task as inProgress

Operator’s job is to verify the details