Cloud Remote Desktop Services

* Basic Architecture

Req →

Game
Video Editing Software
Any High - End Software
Custom Request

→ if App is already specified
⇒ apply the corresponding json file

→ if App is not specified
⇒ Take VM configuration from user

→ Launch the VM in background
↓
Give a browser based rdp using guacd
↓
Users can then choose from their favorite rdp from the browser

* Json structure for the Applications.

- Games
  - → High - End - Games
  - → Mid - , low
  - → Base (for those cases when no application is speified)

- Editing
  - → High - End - Softwares
  - → Mid
  - → low
  - → Base

⋮ Etc.

```
{
  Apps : [        ]

  aws : {
      ami_id : " ___ "
      instance_type : " g4xdn.large "
      Storage - space : " ___ "
  }
  azure : {     }
  gcd :  {     }
     ⋮
}
```

* Request Flow

→ User visits the page → Ping test for user's location to all the data center → User chooses the server. (whether they want to compromise with latency or price) → User provides any extra configurations Ex- Storage, gpu etc.

↓

User pays for the setup → Launch a default RDP on browser → from the browser based rdp user gets to choose whether to change the RDP or not.

User pays for the setup ↙ Razorpay ↘ Stripe

* Status Polling

→ Use cookies and sessions to store user information for polling.

→ Polling will occur every 10 secs to get the status of the VM.

Possible Solution

- Cookies and Sessions with Polling
- Server side push notification after the completion of each stage.

* Timer

→ frontend : Once the VM launches a simple information about startime should be present at frontend. There we can start a stop-watch like timer for UX.

→ Backend : Push all the important information about the VM in a time based queue & terminate the VM once the timer finishes.

Possible Solutions
↳ Job Queue, Message Queue, Task Queue

⇒ User should have ability to increase the timer on the go by paying.
If user pays, the job queue's timer should get updated.

User and Membership Management

* Plans

↳ Hourly Plans : No user data required, should be managed just with sessions & cookies. Doesn't matter even if the user has an account.

↳ Monthly Plans : User account is mandatory. User should be able to see their VMs, running or stopped and all the data should be present there.

$$\begin{tikzcd}
 & \text{Monthly} \arrow[dl] \arrow[dr] & \\
\text{limited VMs} & & \text{Unlimited VMs.}
\end{tikzcd}$$

if idle > 30 mins $\Rightarrow$ Stop the VM automatically to save costs.

user should see their stopped VMs & should be able to start it.

* Automation

1. Instead of launching a default RDP in browser, the game should be launched automatically.

2. Instead of allowing user to choose their favorite RDP ~~&~~ from the browser RDP, allow them to choose beforehand & directly start the required RDP

How to achieve this ?

↳ SSH login to the remote VM
↳ Run a script to execute RDP
↳ fill all the details automatically using python's auto GUI method.

* Database

User
↳ ID
↳ Name
↳ Email
↳ Phone_Number
↳ Subscription_id
↳ User_type
  ↳ Admin
  ↳ Editor
  ↳ client or customer

Applications
↳ ID
↳ Name
↳ Description
↳ Service ID
↳ Application_Pack_ID
↳ Slug
↳

Services/Categories
↳ ID
↳ Name
↳ Description

RDP
↳ Name Ex. Parsec
↳ Description
↳ ID
↳

Can contain
↳ Base types also
base_games.json
base_editing.json

VM_Details
↳ ID
↳ VM ID
↳ Created_At
↳ Public_IP
↳ UserName
↳ Password
or
↳ Encoded_Token
↳ VM_Info (In Json format)
↳ State (Running, Stopped, Deleted)
↳ duration (no. of hours)
↳ Start_time
↳ End_time
↳ Time_Elapsed

only these two are enough to calculate the end date & time

Memberships or Plans/Subscription
↳ ID
↳ Name
↳ Description
↳ Cost
↳ Duration (In Months) ↗ or in Days
↳ Allowed_Application_Packs
⇓
we should not allow all plans Ex- A person purchasing a plan for editing purposes should not pay a high price for a gaming rig
↳ Other required Information like,
↳ no_of_hours/per month
↳ no. of VMs can launch at once
↳

Applications_Pack
↳ ID
↳ config_json
↳ Name ("High-End-Games or some fancy name)
↳ Description
↳

* User_Subscription
↳ user_ID
↳ Subscription_ID
↳ Start_time
↳ Expiry_time
↳ transaction_ID
↳

Cloud Providers
↳ ID
↳ Name (AWS, GCP, Azure, Vultr)
↳ identifier: AWS, Azure, GCP

* Transaction
↳ ID
↳ transaction_id
↳ userd_ID
↳ transaction_amount
↳ created_at
↳ updated_at
↳ Successful
↳ token
↳ Description: which subscription package was chosen, for duration etc.

Application Pack Costing
↳ ID
↳ Application_pack_id
↳ cloud_provider_id
↳ cost } B coz different cloud providers will have different prices for the same VM.

↳ hourly_cost
monthly cost will already be included in monthly subscription

* VM_Info:
  - ↳ Start_time
  - ↳ Duration
  - ↳ Stopped_time → This is not the actual end time but the time at which the instance was stopped. Ex
  - ↳ Elapsed_time

Start_time & stopped time will be changed frequently to calculate the time elapsed.

for original start time, created_At is enough ✓

$$\left.\begin{array}{l}\text{Start time} = 1\,\text{pm}\\ \text{Duration} = 5\,\text{hrs}\\ \text{Stop time} = 2\,\text{pm}\end{array}\right\}\ 1\,\text{hr}$$

Still 4 hrs are left.

Thus Start_time & stop-time can be updated frequently.

* Elapsed_time = Stop-time - Start_time

Every time the same VM is launched, it should also be added in the queue to delete with the scheduled time

= Duration - Elapsed_time