



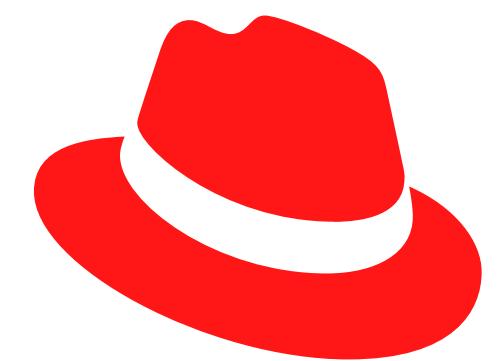
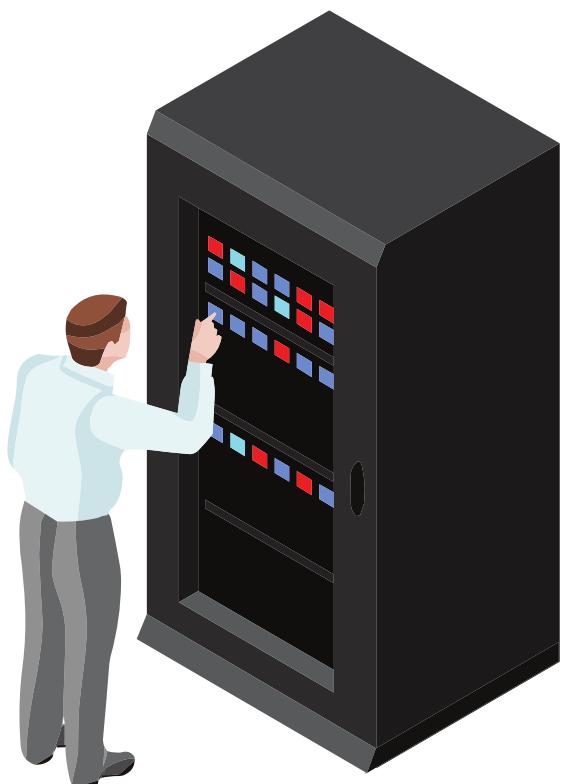
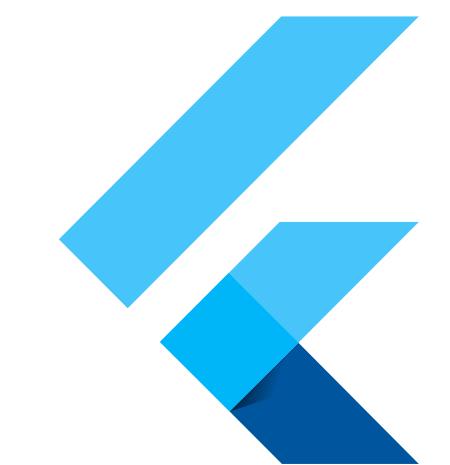
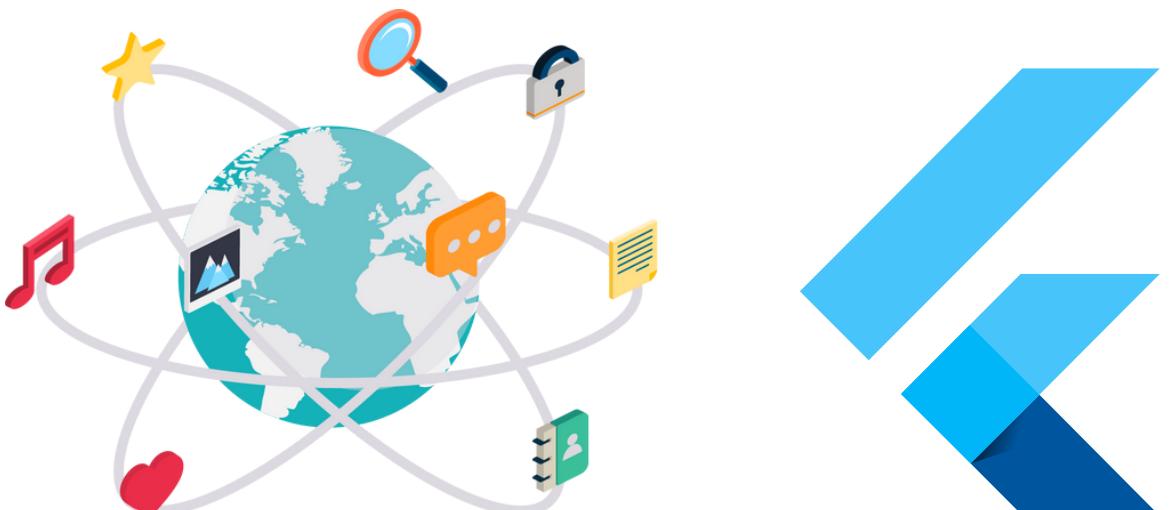
॥ सिद्धः भूषयते विद्याम् ॥

avantika
UNIVERSITY

DESIGN PROJECT | USER
CENTRED DESIGN

TERMINAL
APP

NIPUN PATEL
AU19B1009



Index

Contents	Page Nos.
Abstract	2
Problem Background	3
Domain	4
Identified Problem	4
Problem Research	4
Primary Research.....	4
Secondary Research.....	4
User's Need- Constraints and Challenges (Survey Statistics).....	5
Origin of Research	7
Defining Problem.....	9
Objectives, Project need and Scope.....	9 - 11
Objectives	9
Need of the Project	9
Scope.....	10
Problem Statement.....	11
Ideation Stage	12 - 30
Kano model.....	17
Sketches and concepts	18 - 26
Final concept.....	27
Design Stage	31 - 39
Architecture and flow	32 - 33
Wireframes	35 - 37
Implementation.....	40 - 62
AWS Configuration	41 - 52
Firebase configuration	53 - 54
Frontend Flutter	55 - 62
Testing	63 – 76
Local test cases	64 - 69
Client test cases	70 - 73
Result analysis	74
Limitations.....	75
Conclusion and future scope	76

Abstract

Linux® is an open-source operating system (OS). An operating system is the software that directly manages a system's hardware and resources, like CPU, memory, and storage. The OS sits between applications and hardware and makes the connections between all of your software and the physical resources that do the work. Think about an OS like a car engine. An engine can run on its own, but it becomes a functional car when it's connected with a transmission, axles, and wheels. Without the engine running properly, the rest of the car won't work. Linux was designed to be similar to UNIX, but has evolved to run on a wide variety of hardware from phones to supercomputers. Every Linux-based OS involves the Linux kernel—which manages hardware resources—and a set of software packages that make up the rest of the operating system. The OS includes some common core components, like the GNU tools, among others. These tools give the user a way to manage the resources provided by the kernel, install additional software, configure performance and security settings, and more. All of these tools bundled together make up the functional operating system.

Because Linux is an open-source OS, combinations of software can vary between Linux distributions. The command line is your direct access to a computer. It's where you ask software to perform hardware actions that point-and-click graphical user interfaces (GUIs) simply can't ask. Command lines are available on many operating systems—proprietary or open source. But it's usually associated with Linux, because both command lines and open-source software, together, give users unrestricted access to their computer. Our latest release of Red Hat® Enterprise Linux comes with even more built-in command line capabilities than ever before and includes consoles that bundle those capabilities in easy-to-use modules that exist off of the command line.

Kernel

The base component of the OS. Without it, the OS doesn't work. The kernel manages the system's resources and communicates with the hardware. It's responsible for memory, process, and file management.

System user space

The administrative layer for system-level tasks like configuration and software install. This includes the shell, or command line, daemons, processes that run in the background, and the desktop environment.

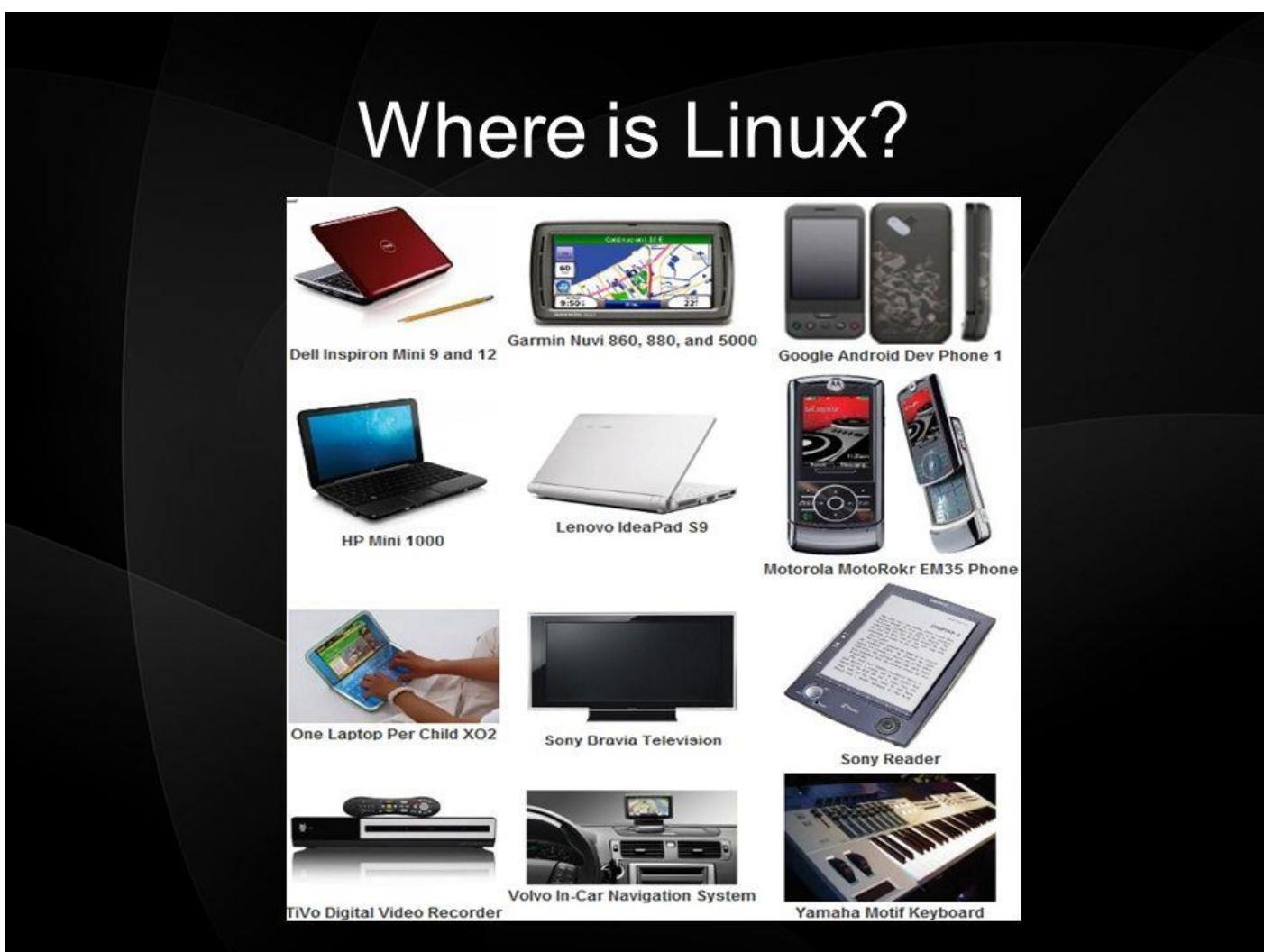
Applications

A type of software that lets you perform a task. Apps include everything from desktop tools and programming languages to multiuser business suites. Most Linux distributions offer a central database to search for and download additional apps.

Linux is a free, open-source operating system, released under the GNU General Public License (GPL). Anyone can run, study, modify, and redistribute the source code, or even sell copies of their modified code, as long as they do so under the same license.

Linux has become the largest open sources software project in the world. Professional and hobbyist programmers and developers from around the world contribute to the Linux kernel, adding features, finding and fixing bugs and security flaws, live patching, and providing new ideas—all while sharing their contributions back to the community.

| Problem Background



Linux is a Virus Free Operating System as it is open source but lacks in GUI, However most of the top companies prefer using the Linux based Operating system due to safety reasons and data breaches, but the thing is there is less platforms where Linux based operating system can be seen for an individual user, so if you work for some organization who is using Linux , you lacks the knowledge there as we have habit of using windows based GUI OS and we feel discomfort in using that initially, if you are a teaching organization you will find dedicated system to practice but if as an individual you won't prefer virtual machines to perform a quick query, so where is Linux.....?

❖ Domain Selection:

Education.

❖ Title:

Fun with Linux

❖ Identified Problem:

The current problem deals with limited no. dedicated system of Linux only for users who are on first come first serve basis, in the world of 5G with advance technology there is no such online compilers for Linux for use on the go or practice some particular commands for the user, even amazon EC2 instance are chargeable or we use virtual machine which slow down the performance of PC.

❖ Problem Research:

Primary Research:

Reading books and articles also gone through the research articles.

Below is the research work, which tells about the existing solutions and current challenges faced and the future of Linux as a cloud service.

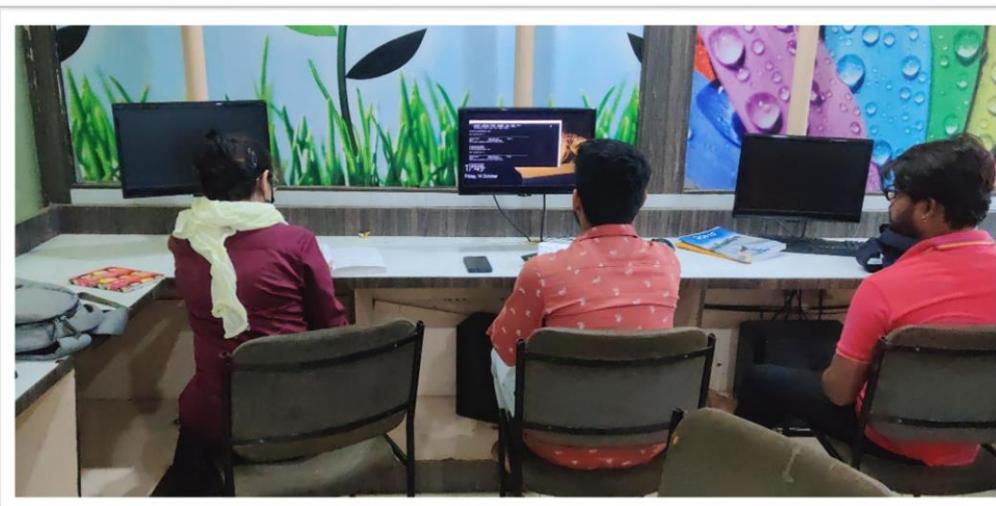
[Research Paper 1](#) | [Amazon AWS PDF](#) | [Reliance digital guide](#)

Secondary Research:

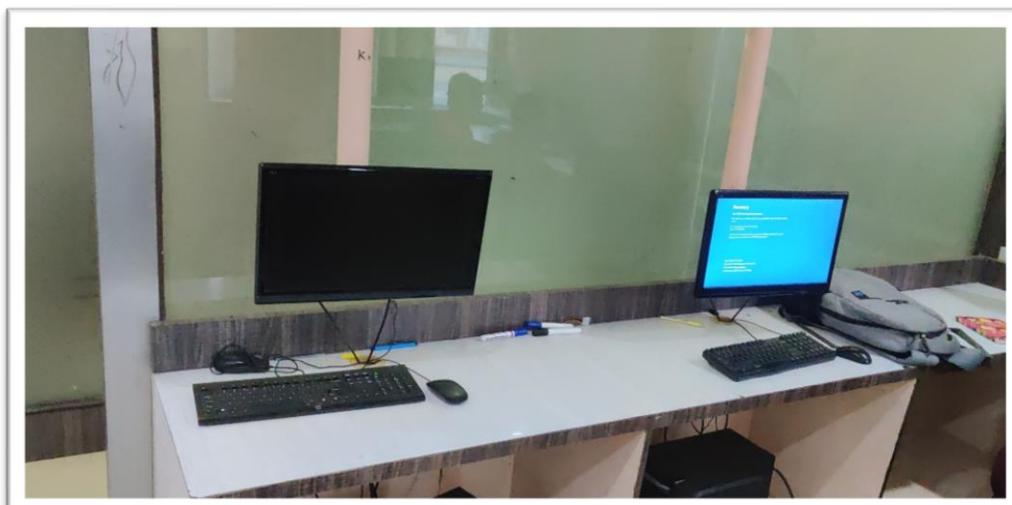
Asking different users about the Linux, usage and taking feedback for the same, getting inputs from our expert teachers and understanding their requirement.

❖ Origin of the research work:

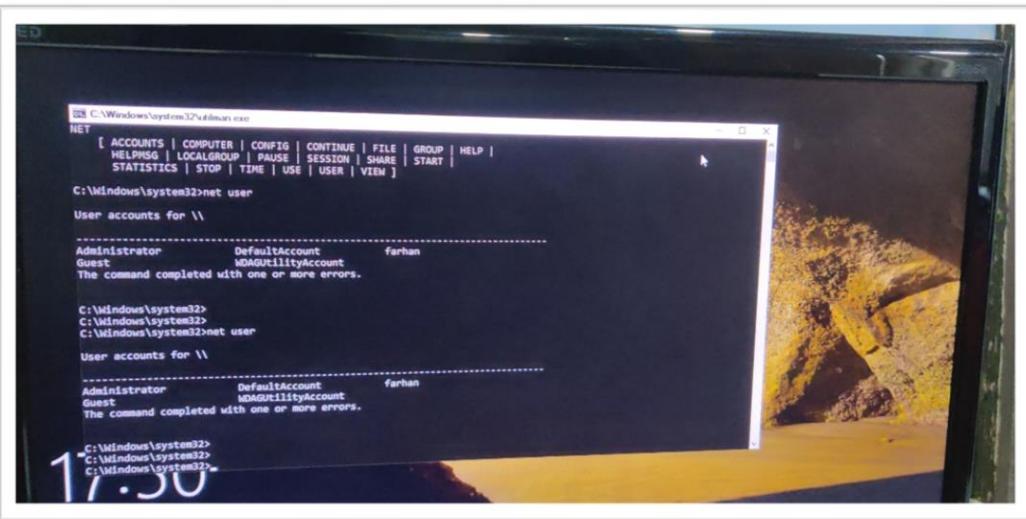
We have visited an Institute named as **IANT (Institute of Advanced Network and Technology)**. Where we saw students coming from different parts of the region are preparing for different exam such as RHCSA, RHCE, RHCA, CompTIA N+ etc., the total no. of students enrolled for the courses are more than 160 and the no. of dedicated systems to practice are not more than 50, so the rest 110 students practice in different time slots which is highly time consuming and also some student work on their personal Pc's with virtual machine which reduces their system performance but the remaining candidates don't have the systems so they are totally dependent.



Students practicing commands on Linux dedicated PC's



GUI based PC's runs virtual machine for Linux



A screenshot of a Windows command prompt window titled 'C:\Windows\system32\cmd.exe'. The window displays the output of the 'net user' command twice. The first execution creates a new user account named 'farhan' with 'Administrator' privileges and sets it as the 'DefaultAccount'. The second execution shows the account has been successfully created. The command prompt also lists 'Administrator' and 'Guest' accounts.

```
C:\Windows\system32>net user
User accounts for \\\

Administrator          DefaultAccount          farhan
Guest                  nt AUTHORITY\Account

The command completed with one or more errors.

C:\Windows\system32>
C:\Windows\system32>
C:\Windows\system32>net user
User accounts for \\\

Administrator          DefaultAccount          farhan
Guest                  nt AUTHORITY\Account

The command completed with one or more errors.

C:\Windows\system32>
C:\Windows\system32>
C:\Windows\system32>
```

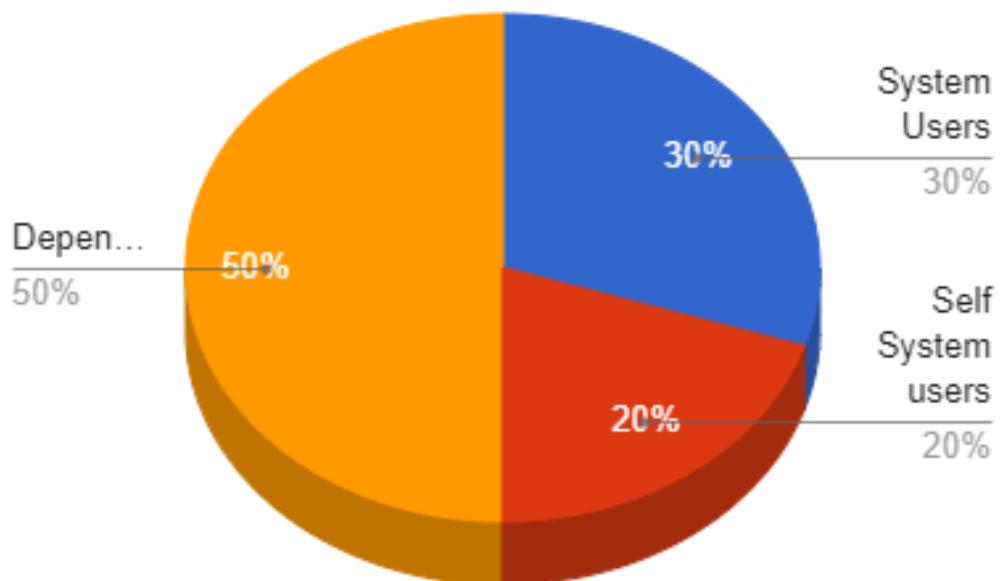
Creating an Apache Server on Red hat RHEL 8.0



Interaction with Instructor Mr. Tausif Khan Sir

❖ Statistics:

Distribution of Linux users via different channels

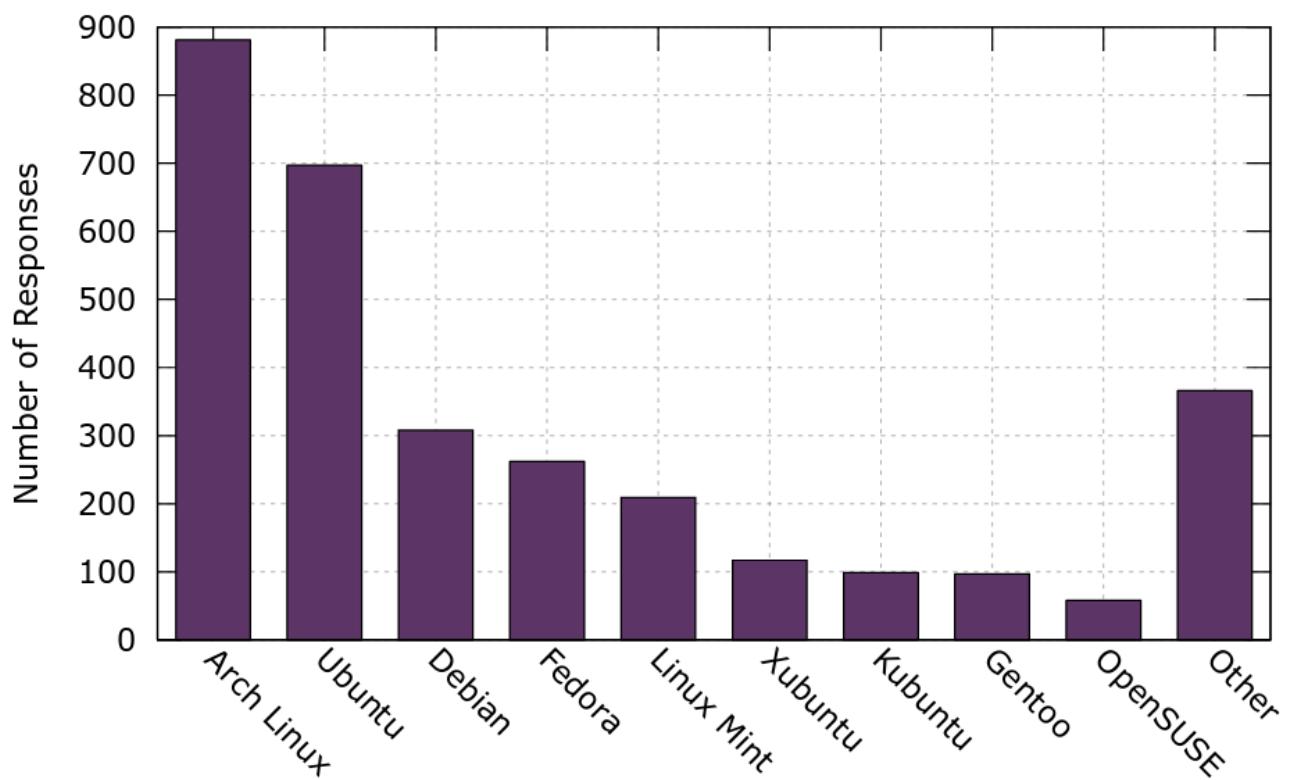


Distribution of users in IANT using Linux via different channels



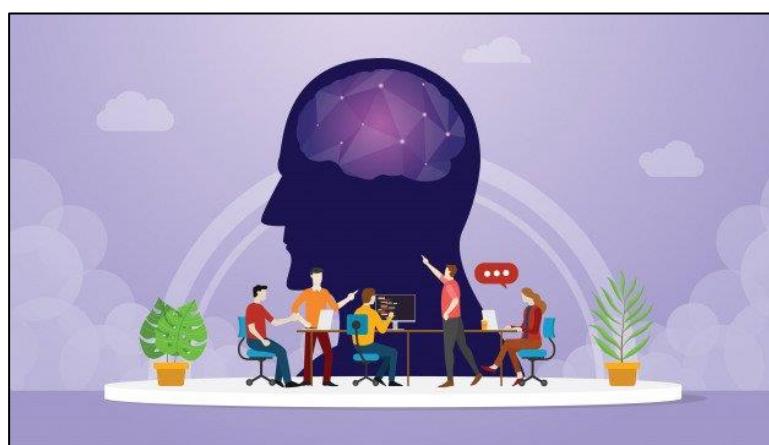
Top Organizations uses Linux operated systems for their various services.

Which Linux distribution do you primarily use on your desktop computers?



Distribution of Different Linux OS mostly used worldwide

❖ Current Challenges faced:



1. Providing timeslots to practice for limited time period.
2. Scheduling alternate day classes may lead to revision loses.
3. Student totally dependent on Institute needs to be given priority.
4. Course content need to revised by performing practical's, no portable solution is there.
5. Linux online compilers are available only for bash scripts not Realtime creations.

❖ Defining the problem: -

After going step by step procedure from understanding the user's need to key learnings from the research papers, articles and our own understanding about this domain we came upon the conclusion of what actually the problem is which is likely to be as –

"Human in today's world likely to use plug and play things or make them simpler rather giving a manual effort so, problem is online Linux compilers exists but not with public usage or you need to operate over organizations cloud service by some amount."

❖ Objectives, Project need and Scope: -

Objectives:

1. To Create a Linux OS based server which is shared publicly.
2. To create a convenient UI for accessing the server via different platforms.
3. To enhance the safety features of users by providing limited storage and permissions.

❖ Why this problem matters/ need of this project:

1. Education Institute's, big organizations, cloud players require dedicated Linux operating employees, if we have limited system inventory will arose the concept of time slots which is a hectic process so in order to save the time and increase the performance optimization is required.



2. Flexibility in accessing is designed to improve the quality of education in interior architecture. Since education is a right and a luxury, everyone needs it.
3. Portable accessing is a flexible method can increase performance and improve the quality of practice by serving several functions at once.
4. Need of flexible access which could act programmatically and balance spaces in terms of performance and efficiency.

❖ Scope of the project:

1. Building an access portable with the help of certain SDK's and reducing manual efforts by enabling the digital means of conduct.



2. Enhancing UI for the Linux access portal ensuring better user experience.
3. Interswitch able platforms from one device to another in terms of cloud medium done programmatically.
4. Enhanced safety feature, applying restrictions and permissions to the user.
5. Optimization of the server handling user load on accessing the it via various channels.

❖ **Problem Statement: -**



Creating an optimized Linux server access portal for different platforms which executes and send commands programmatically by means of interconnected gateway.



STAGE 2

IDEATION



25% on progress...



IN THIS STAGE

We will discuss the hierarchy and procedure of ideation from defining the sketches and generating solutions and in the end concept finalization

01

REQUIREMENTS

02

MINDMAPPING

03

KANO MODEL

04

PERSONA

05

SKETCHES

06

FINALIZATION



USER REQUIREMENTS



While understanding the problem there are certain requirements that are necessary to be fulfilled for any kind of solution provided to the user for the application which we are working i.e.

- ✓ Signup with terminal credentials only
- ✓ Mockup tests to be secured
- ✓ Convenient UI for terminal access
- ✓ Hassle-free terminal accessibility
- ✓ ACL for every user for security
- ✓ Time based and score feature in tests

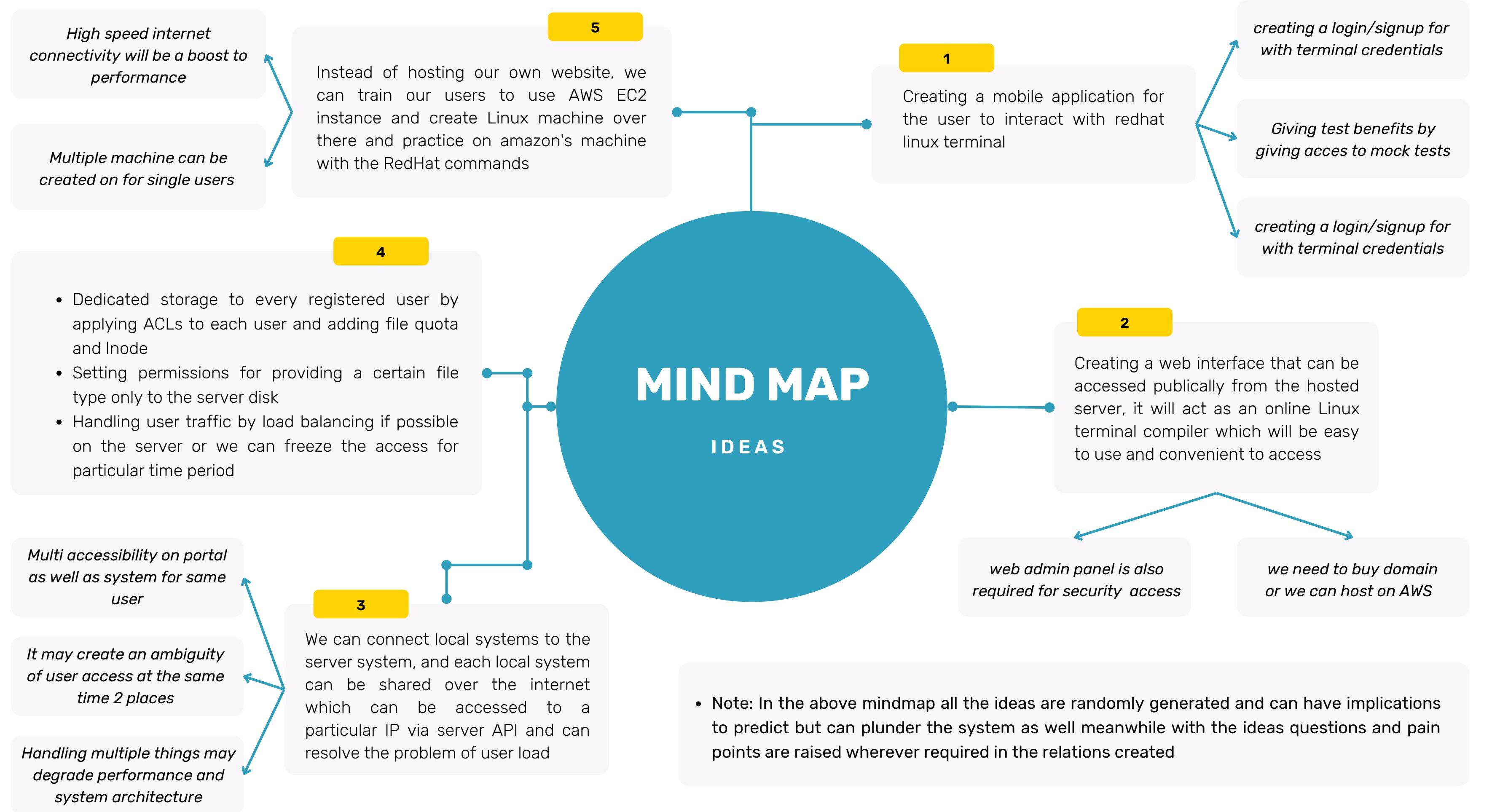
FUN WITH LINUX

USER PERSONA



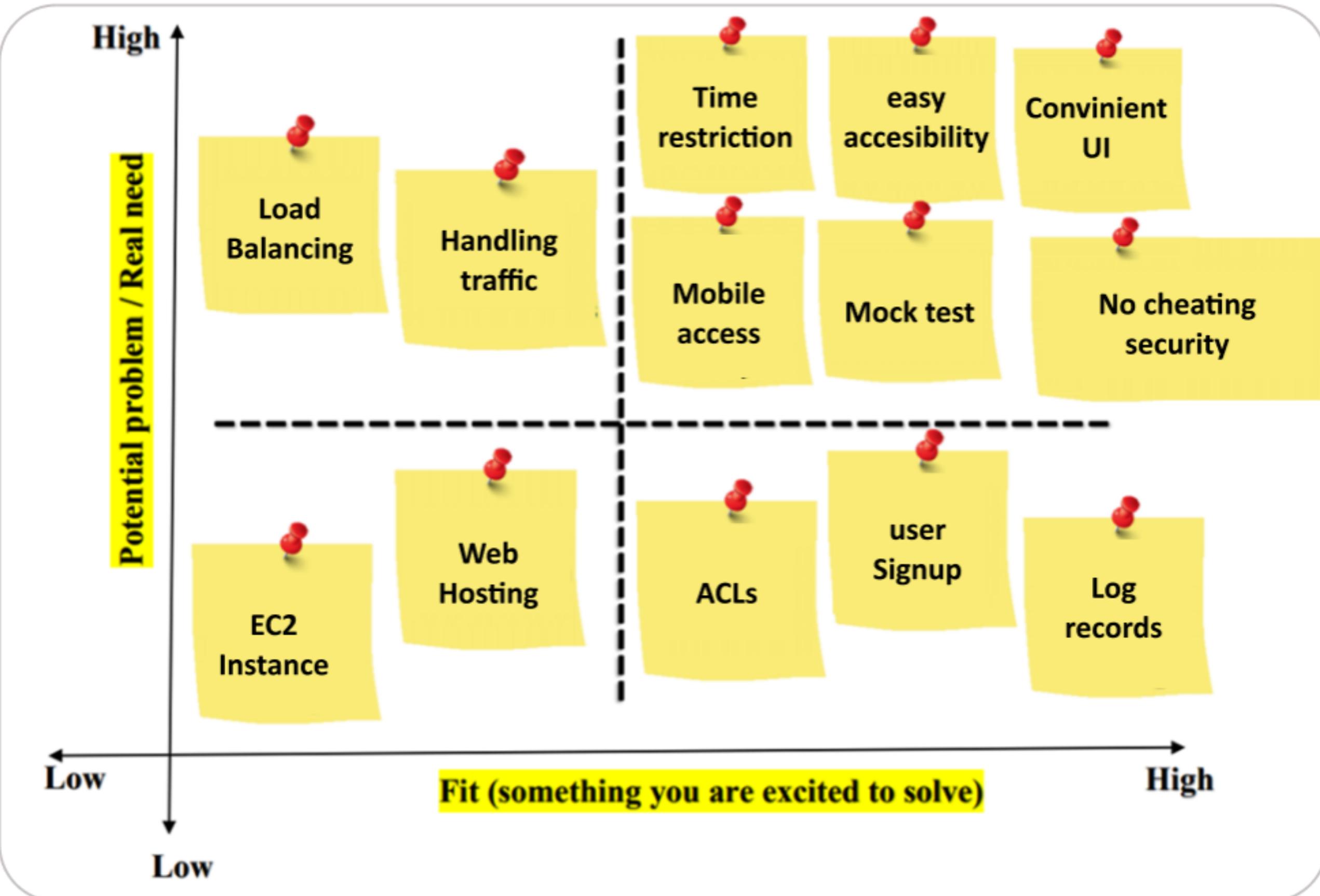
NAME	Akshat Banna
AGE	21
LOCATION	Tarana
OCCUPATION	Network Administrator

INTERESTS	Travelling, cooking, sports, reading books.
CHALLENGES	Spends lots of time in traveling as he lives far away and didn't get time to practice much and focus on the exam paper leading to score less
STEPS TAKES TO FIX THAT	Reducing the time of travel and increasing the practicing time can help a lot to him
GOALS	Hardworking person and highly skilled in coding, his goal is to move from his current location secure a good job and reside in a sophisticated society with his family
MOTIVATION	Practicing and coding
FRUSTRATION	Reminding the commands for creating long type servers is a painful job for him
SOURCES OF INFO	Institue of Advance Network and Technology (IANT), Ujjain (M.P.)

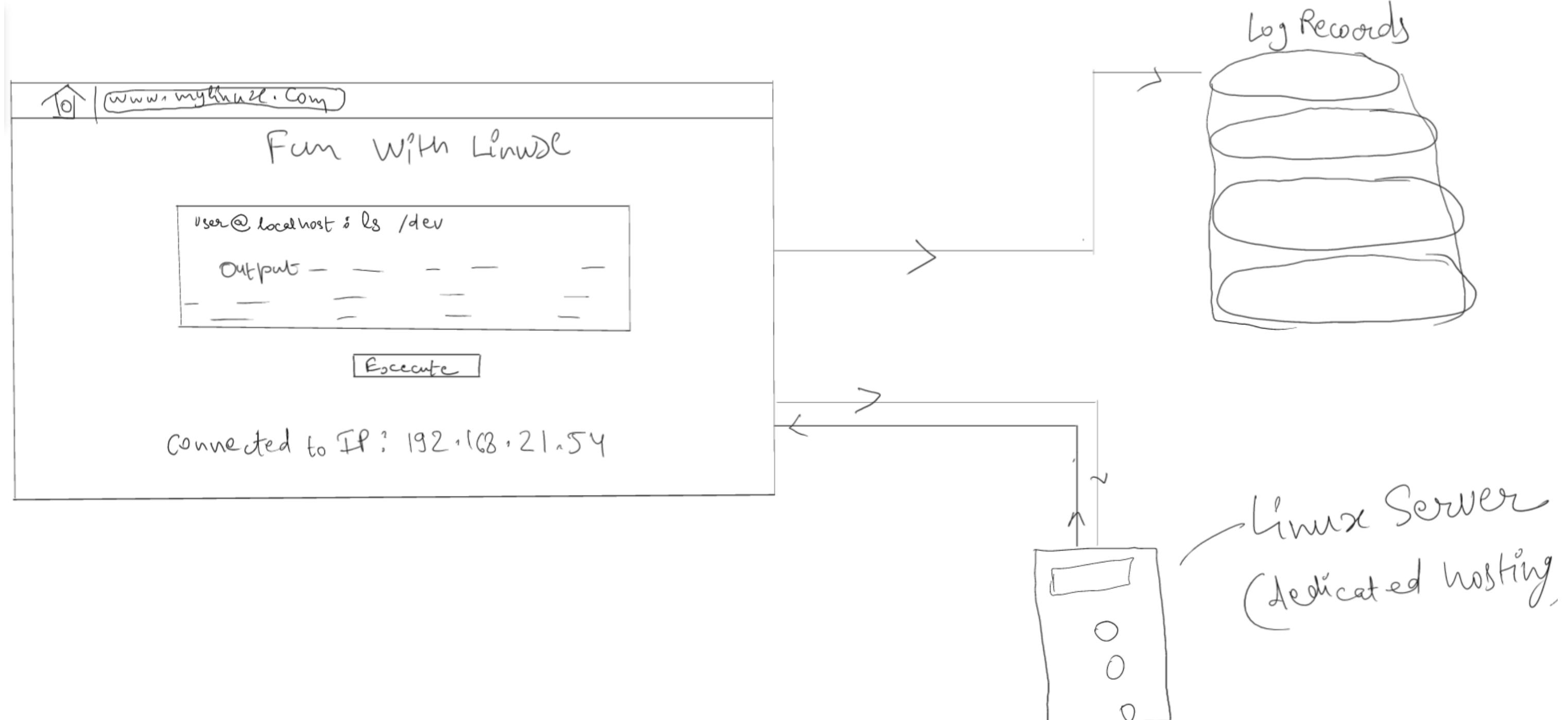


KANO MODEL ANALYSIS

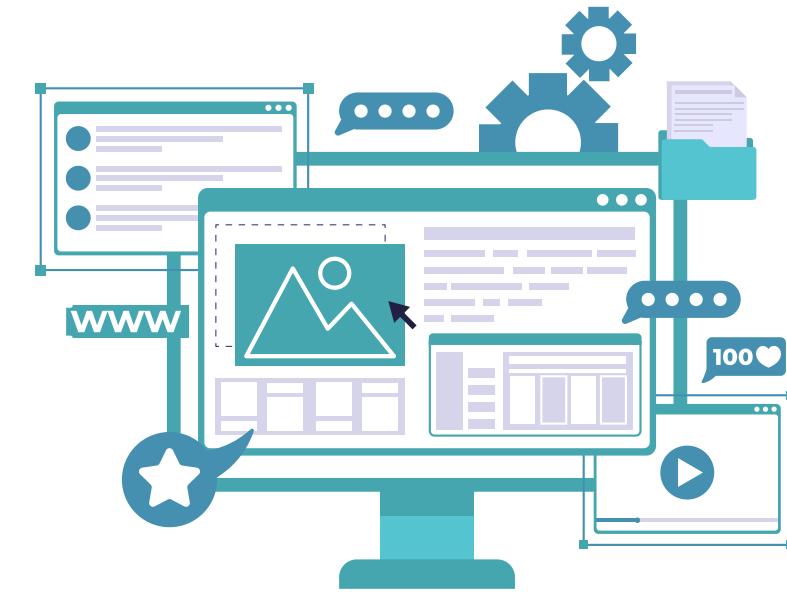
In the given image we have divided the mind map ideas and user requirements into four quadrants of feasibility and will execute the most required need and the highly feasible delighter



SKETCHES AND CONCEPTS



Web Service



In this, we can create a webpage that has a convenient UI for user understanding and is given public access in order to use as a terminal compiler for practicing in real-time, at the backend it will be created using SQL which will store the log record from a particular I.P. and the Linux server will act as a moderator in the execution of commands

Web Service

PROS

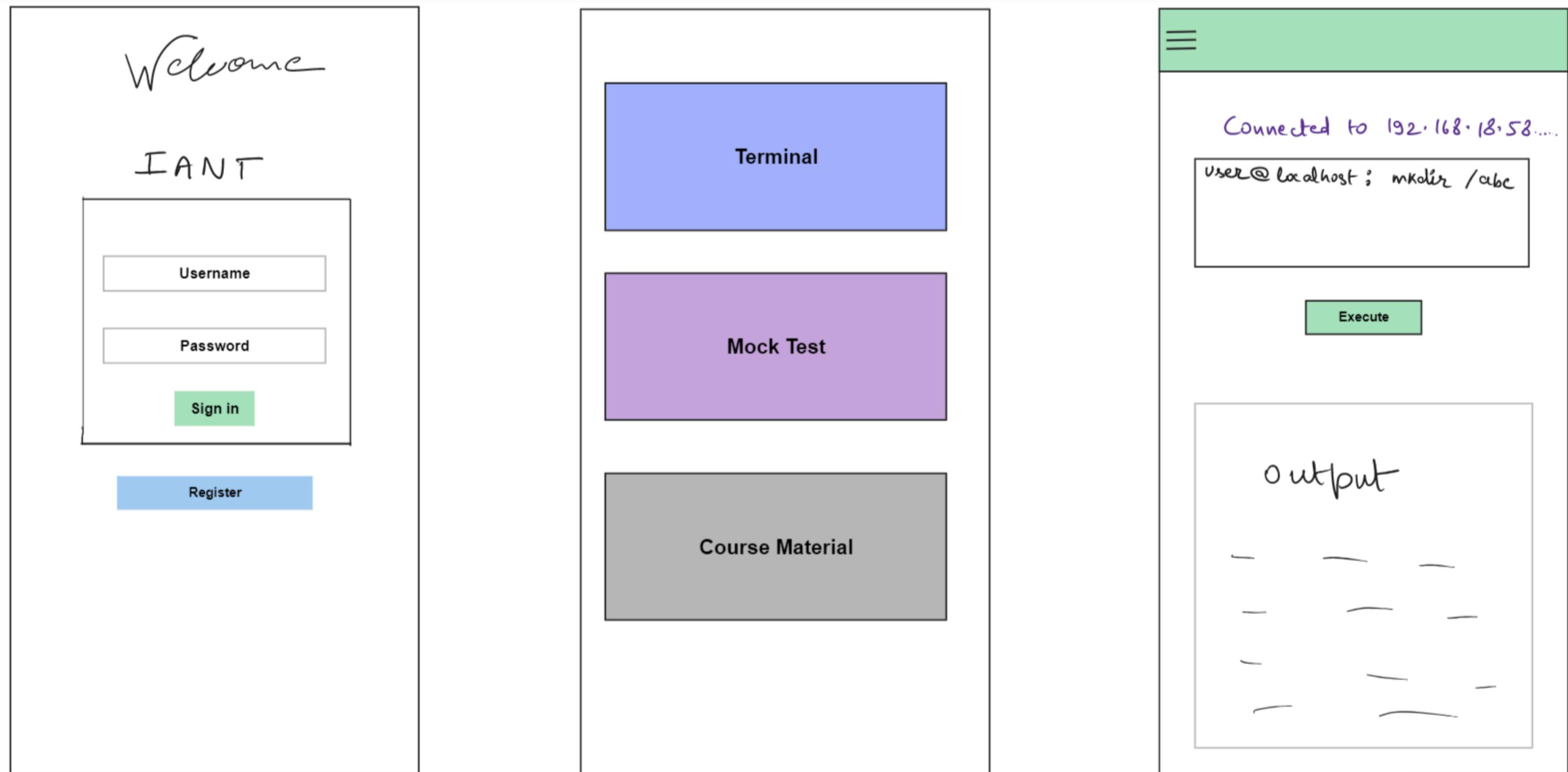
- On our server, dedicated self-hosting makes access simple.
- Convenient UI for easier understandability of terminal
- Direct access to the compiler saves the time of the user
- A high-performance server system and suitable high-speed internet are an ideal match.
- Realtime accessibility on any device at any time is a game changer

CONS

- Fear of getting hacked as it is web-hosted
- No test practice session is available for use at the terminal
- No Proctoring or firewall to monitor the command activity may kill or halt the system
- The user traffic and accessing simultaneously may affect load balancing on the server
- Lack of features limits to act as compiler usage only

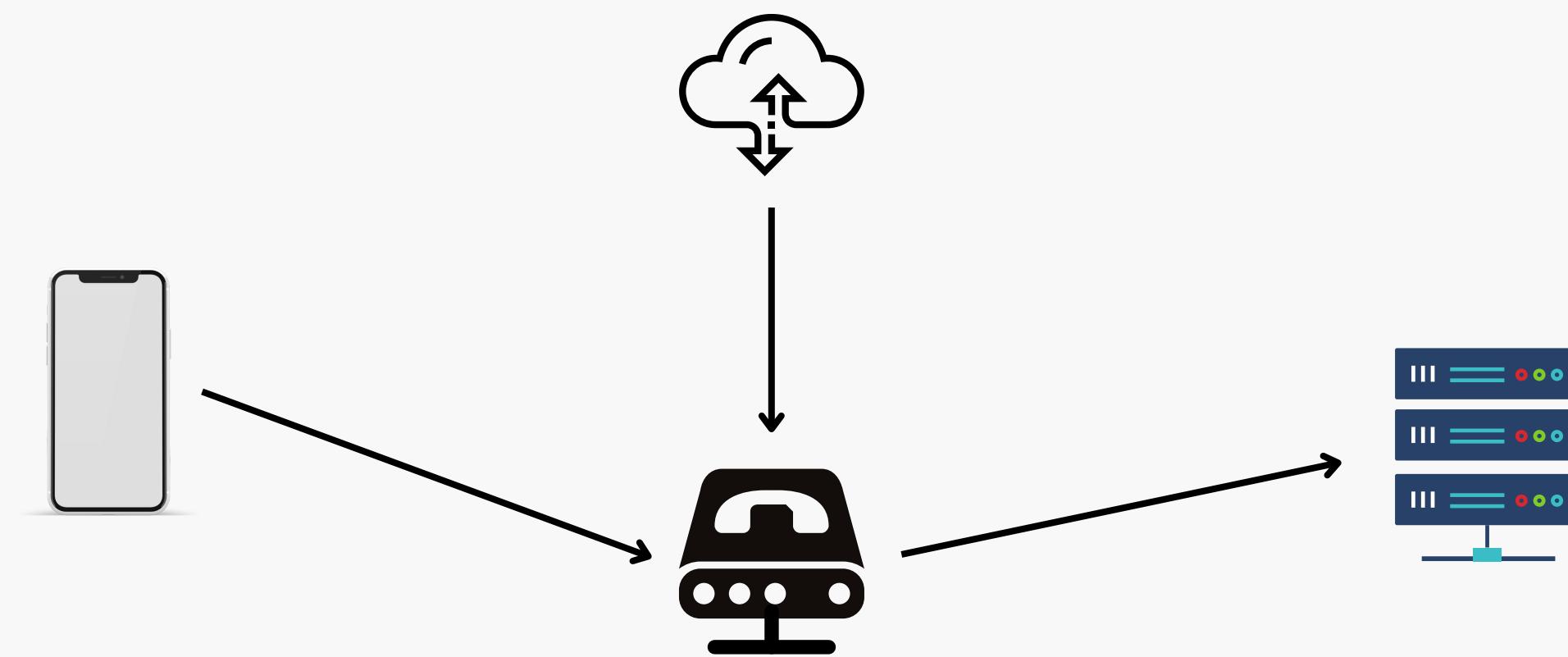
App Service

02





In this, we will be creating an app with an interface connecting to the backend Linux server which executes the terminal command while for the storage, firebase will be the best example to consider



App Service

PROS

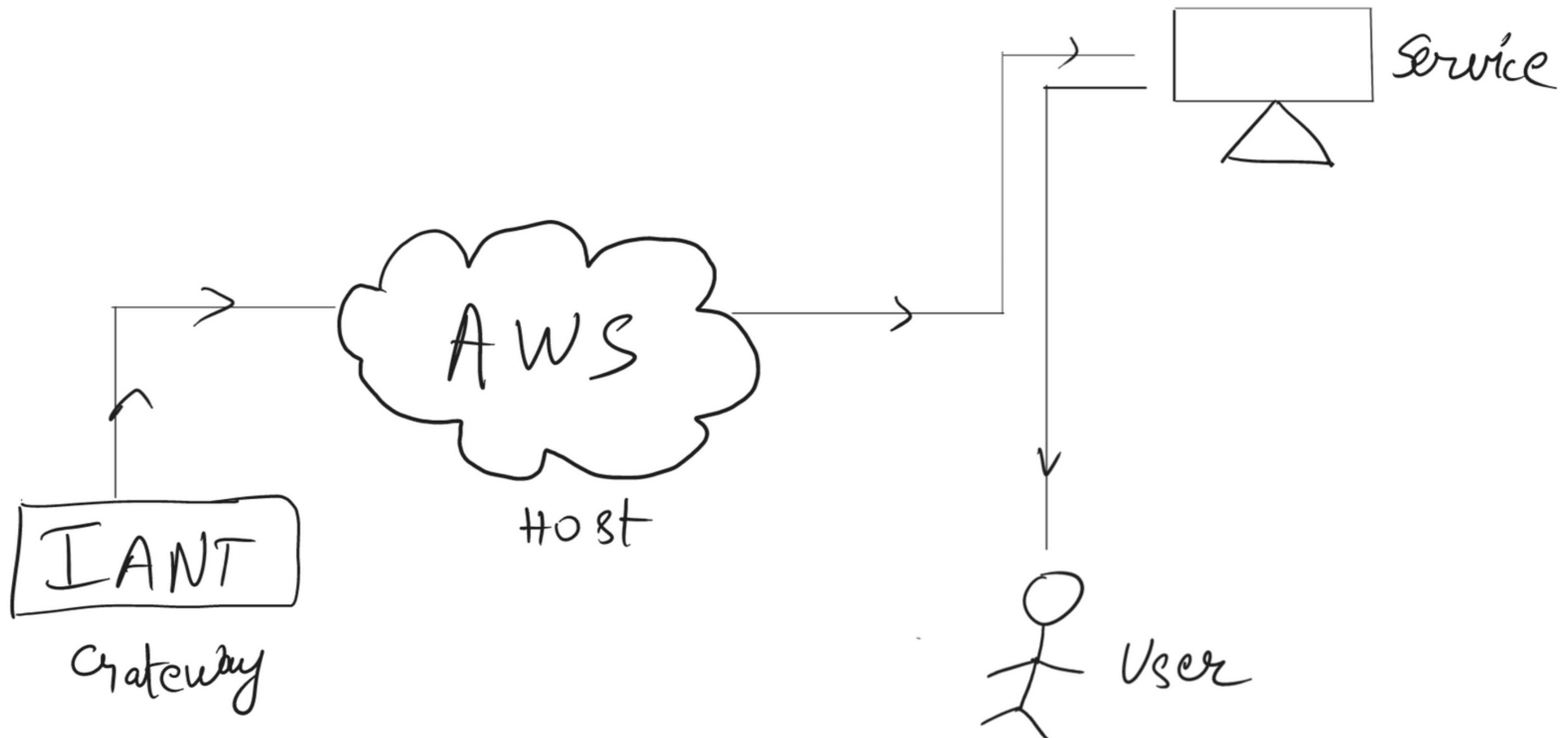
- Support multiple features which is a benefit for the user
- Convenient UI for easier understandability of terminal
- A cost-effective solution, easily accessible to everyone
- Private access with dedicated storage is a boom
- Dedicated institute support helps in growing the sector
- Improves performance and reduced time for accessibility

CONS

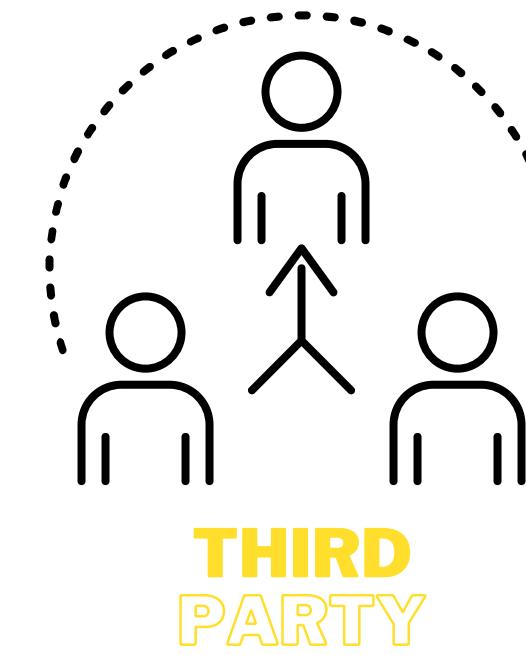
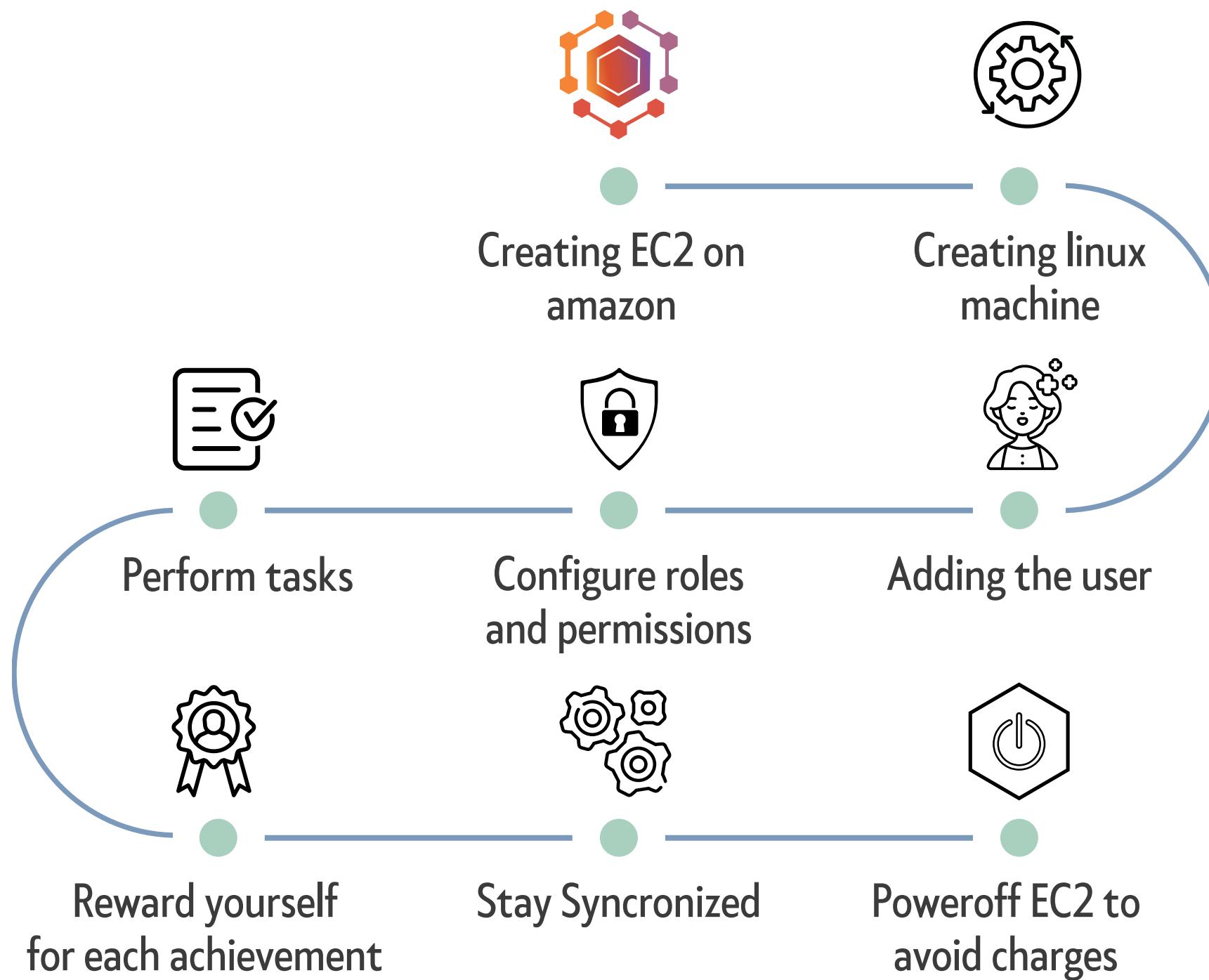
- Avoiding cheating is a concern of doubt
- Limited screen size can clutter the workspace
- Scalability and load balancing is a concern

Cloud Service

03



Stepwise Procedure



In this, we will be using AWS cloud to use Linux terminal service it will be a third-party involvement and application for organizations may or may not consider this criterion as it may have used data breach concerns but a detailed procedure is cited on how to configure for organization users and use it anywhere on any device and any time accessible

Cloud Service

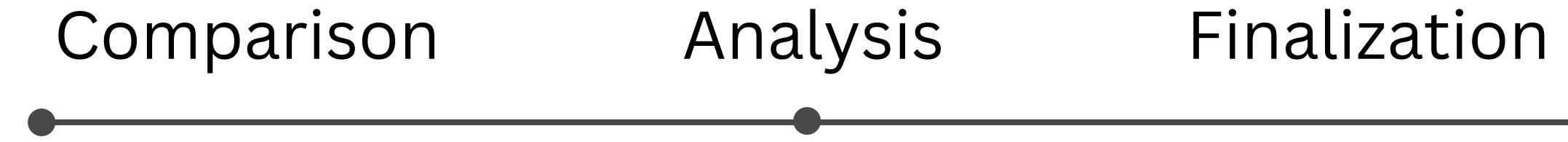
PROS

- Super Speed in terms of performance is granted
- Multiple machines and terminals can be created for a particular user

CONS

- Requires a high-speed active Internet connection
- The facility is chargeable to use on basis of hours
- Need to understand the interface before using it
- The creation of machines can be a complicated process
- Using a third-party app will not monitor the user and can be an unreliable source as it may breach data

Final Concept



01

The web interface can help the users access from anywhere anytime but its features to restricts users to using the terminal as a compiler service only, no test sessions are seen to be delivered, also public hosting for organizations doesn't seem to make a profit or any earning for them

Impact**02**

The mobile interface serves an impactful decision as it serves the terminal access, mock practices as well as delighted services to monitor the user command history, thus it only lacks the service of scalability and load balancing with limited screen space to work upon

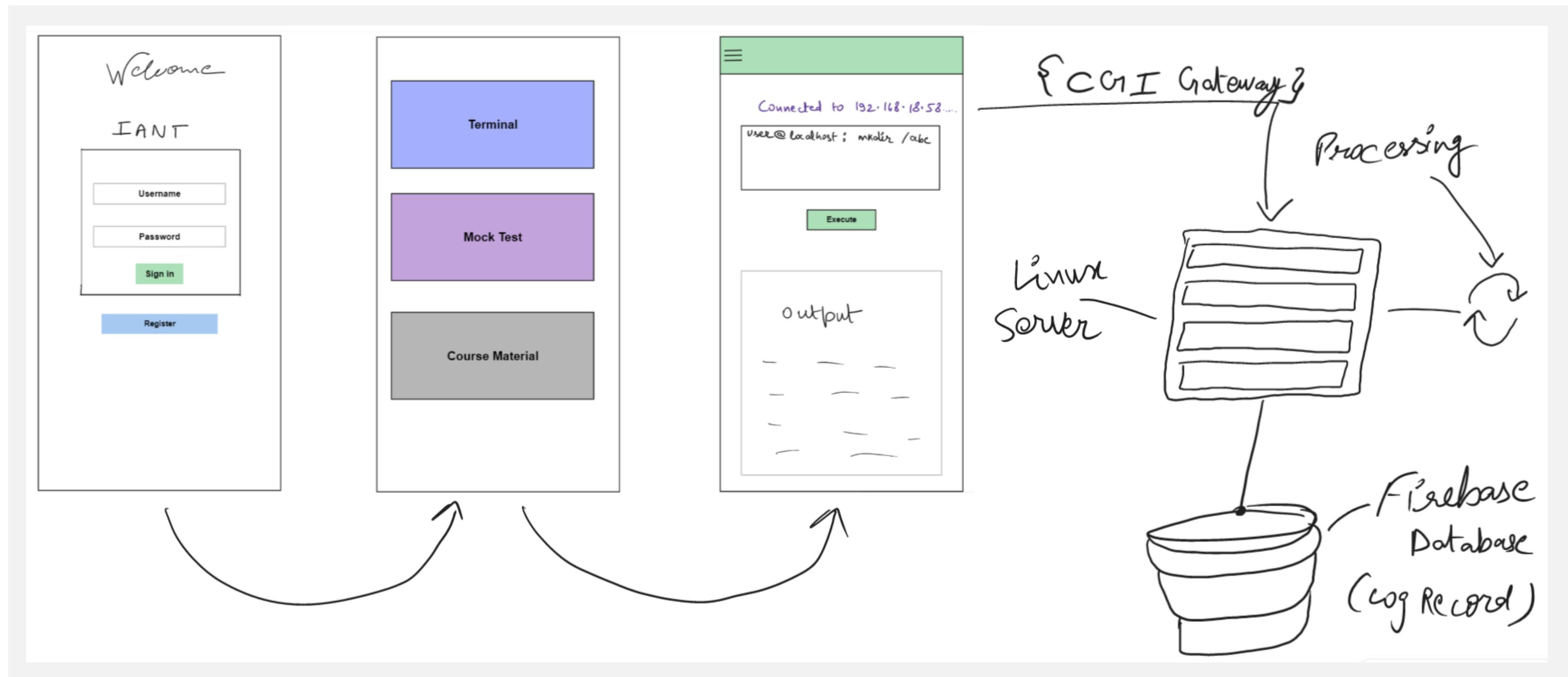
Impact**03**

The AWS service is also a good solution but it involves a third party working on their service which may raise the organization's data breach concern, also connectivity is a line of sight in this case, as well as pricing, occurs for using the service and understanding the interface of one go is a painful game.

Impact



FINALIZATION



MOBILE SERVICE

So on deep analysis and comparison, we came upon finalizing the idea to move ahead with mobile service as this service is easily reachable and affordable to users, there are many features with limited constraints and some of the main key features are mentioned the right

However, there are constraints but users can have easy access with the organization's controlled monitoring of it.

Terminal Service

This feature allows to use but on a limited screen size

Mock Tests

User can take proctored test with time limit constraint

User Roles and permissions

Admin can apply dedicated permissions to each registered user



STAGE 3

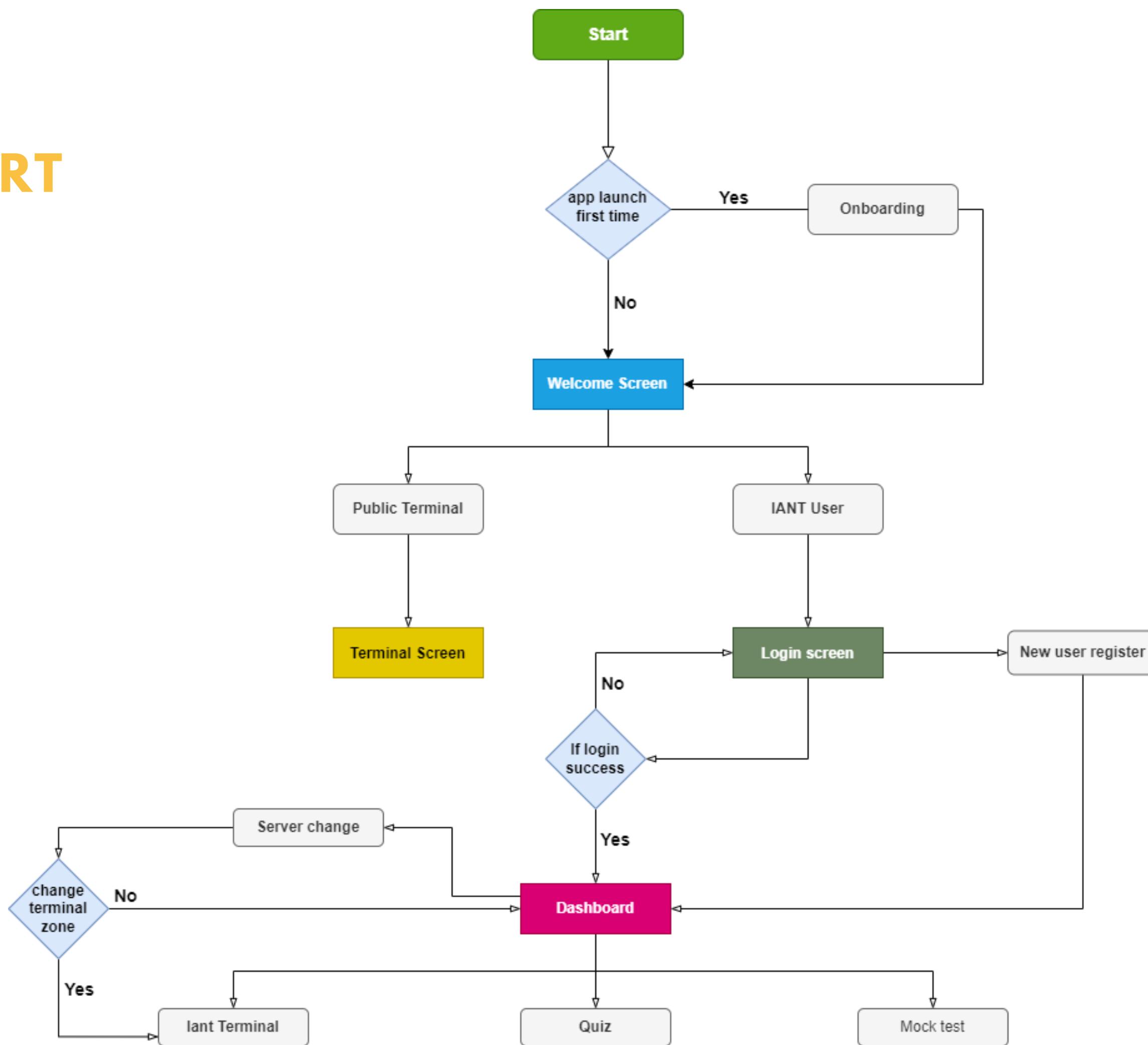
DESIGN



40% on progress...



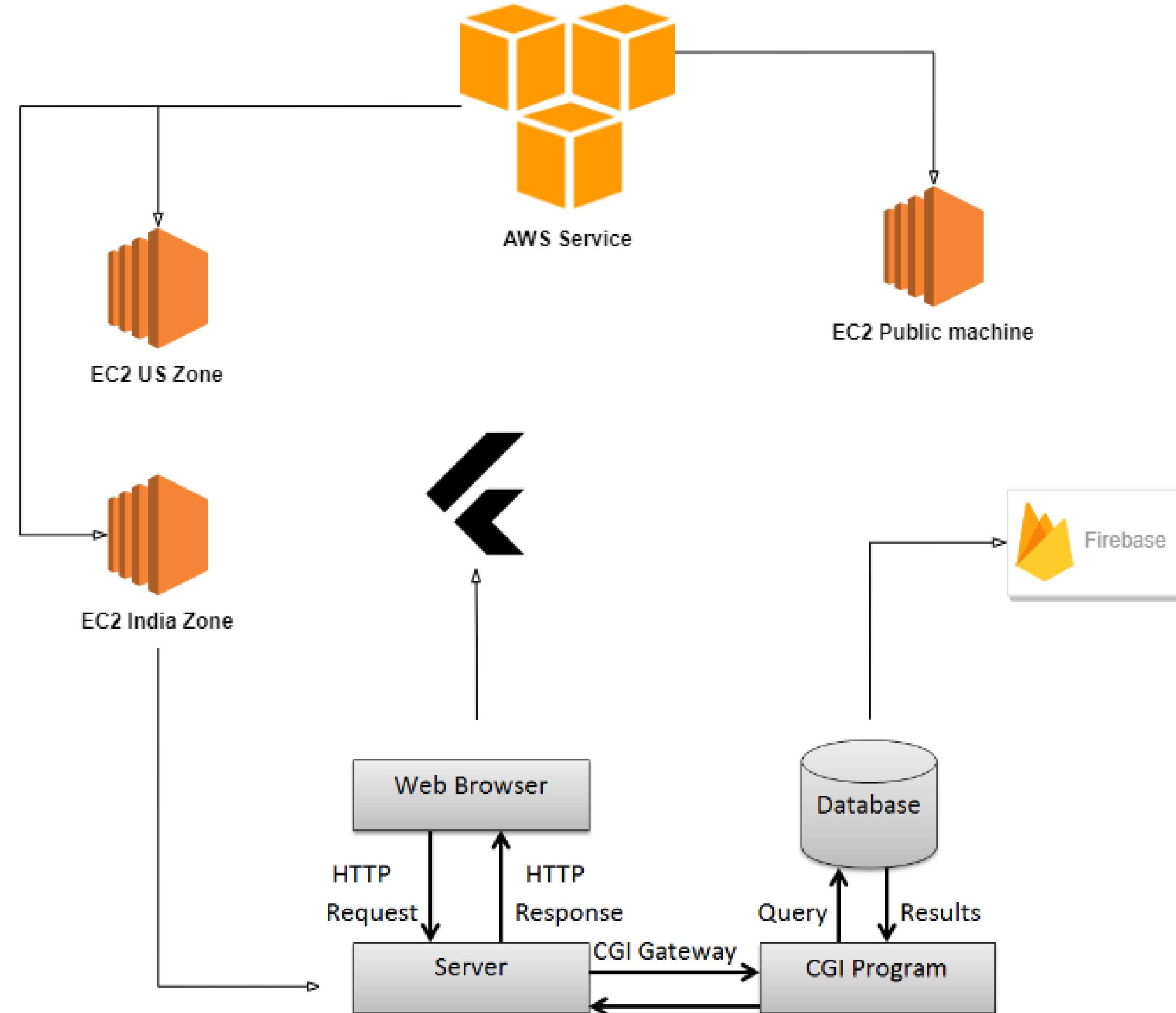
FLOWCHART



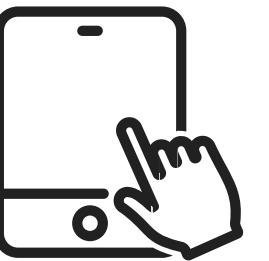
ARCHITECTURE

Common Gateway Interface (CGI)

is an interface specification that enables web servers to execute an external program, typically to process user requests. Such programs are often written in a scripting language and are commonly referred to as CGI scripts, but they may include compiled programs.



FUNCTIONAL REQUIREMENTS



Portable availability

We need to make the platform available in hand go, i.e. accessible 24/7 anytime from the application



Quiz practices

Time-based control in each quiz must be a fair delivery



Request handling

The HTTP requests must be handled as they will be the backbone for the command delivery



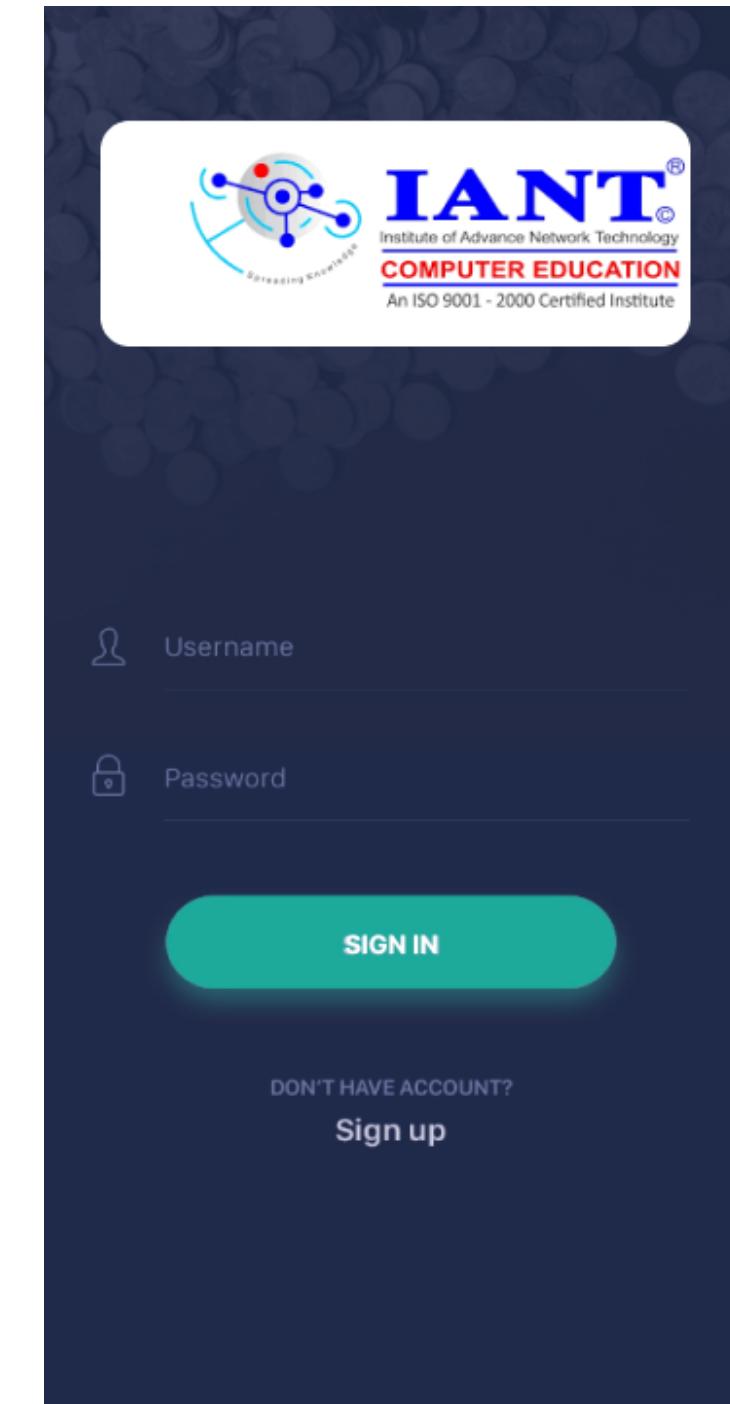
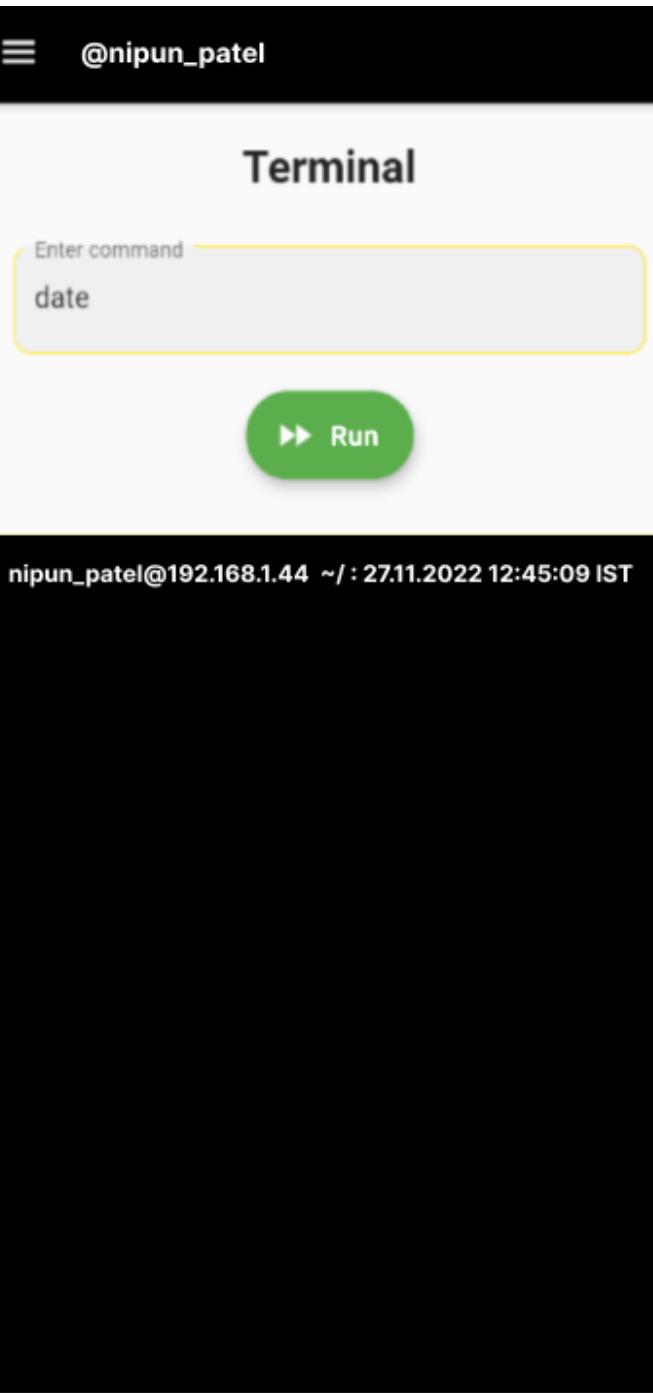
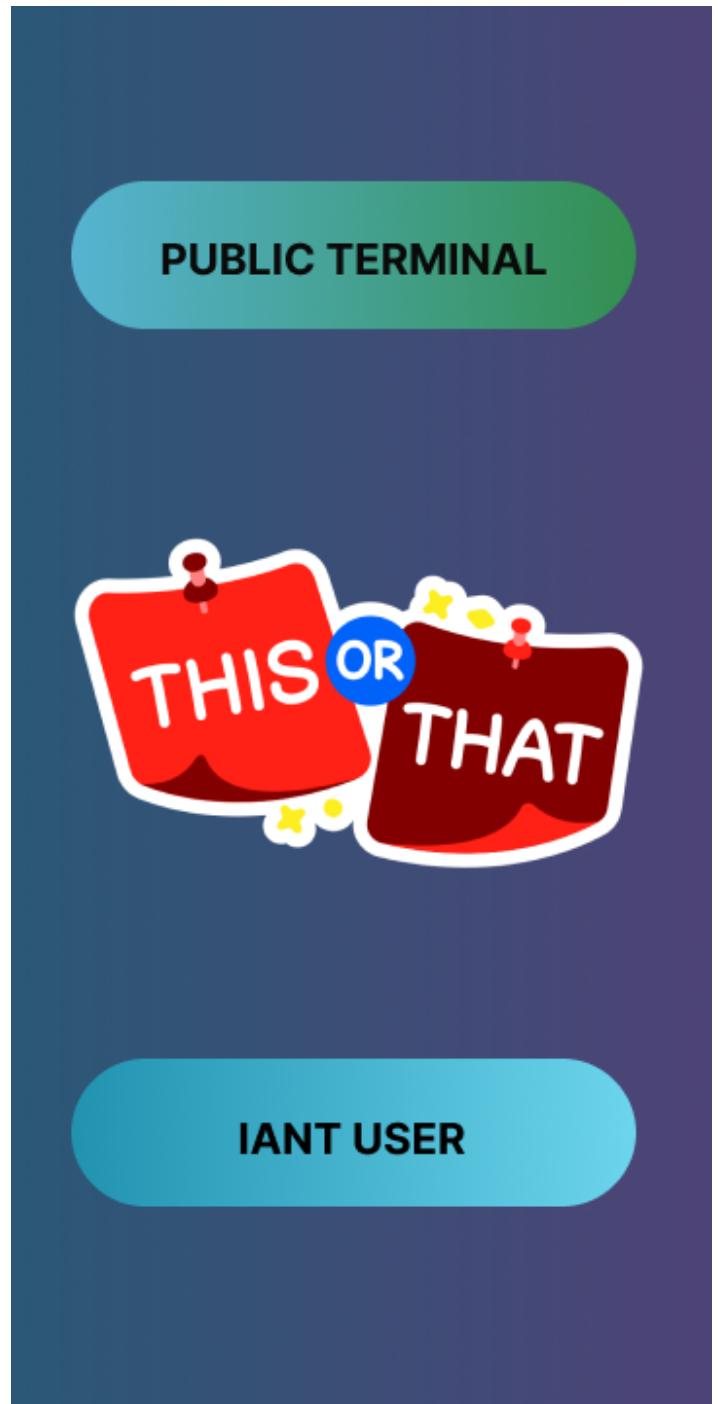
EC2 Scripting

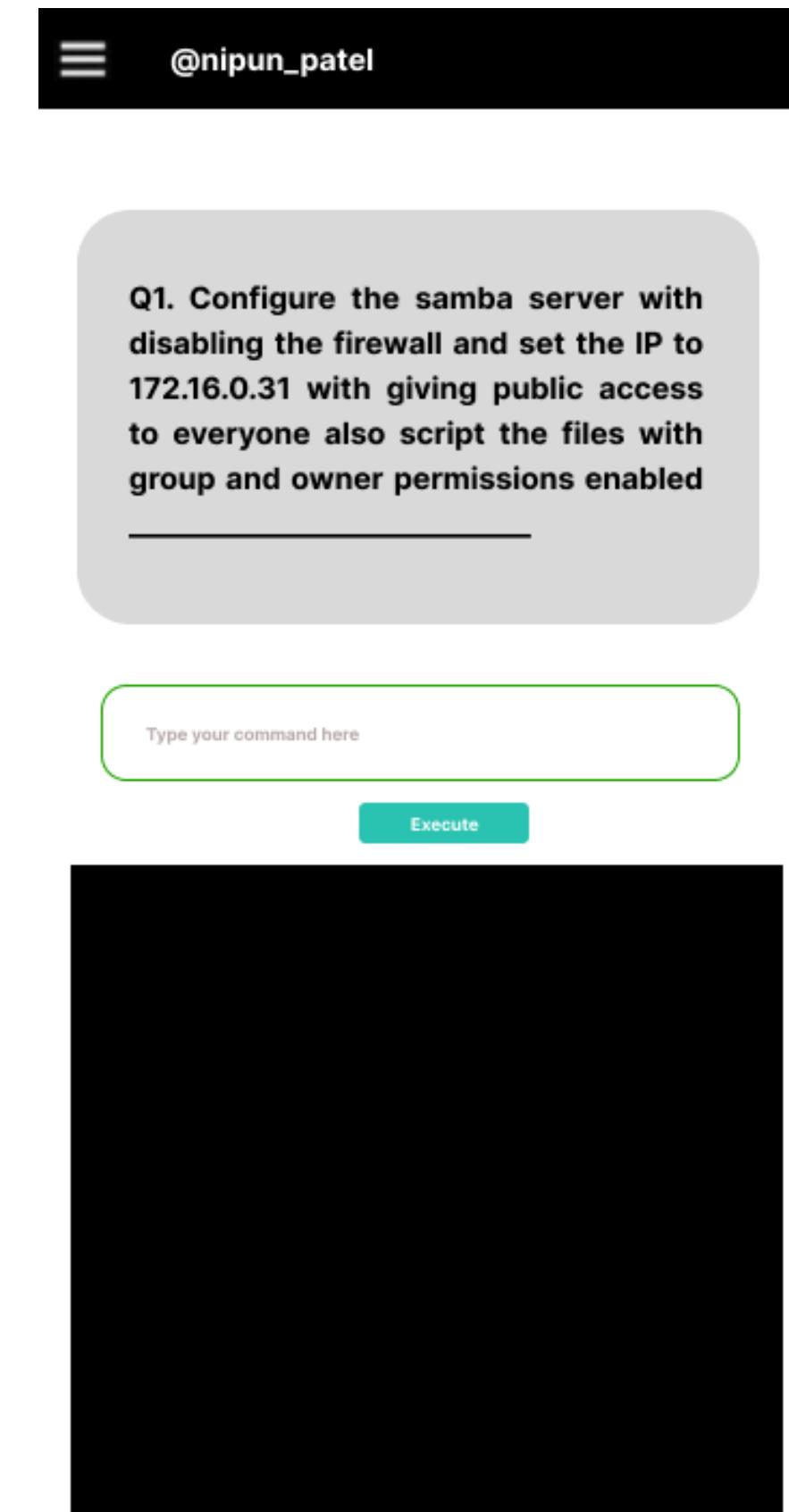
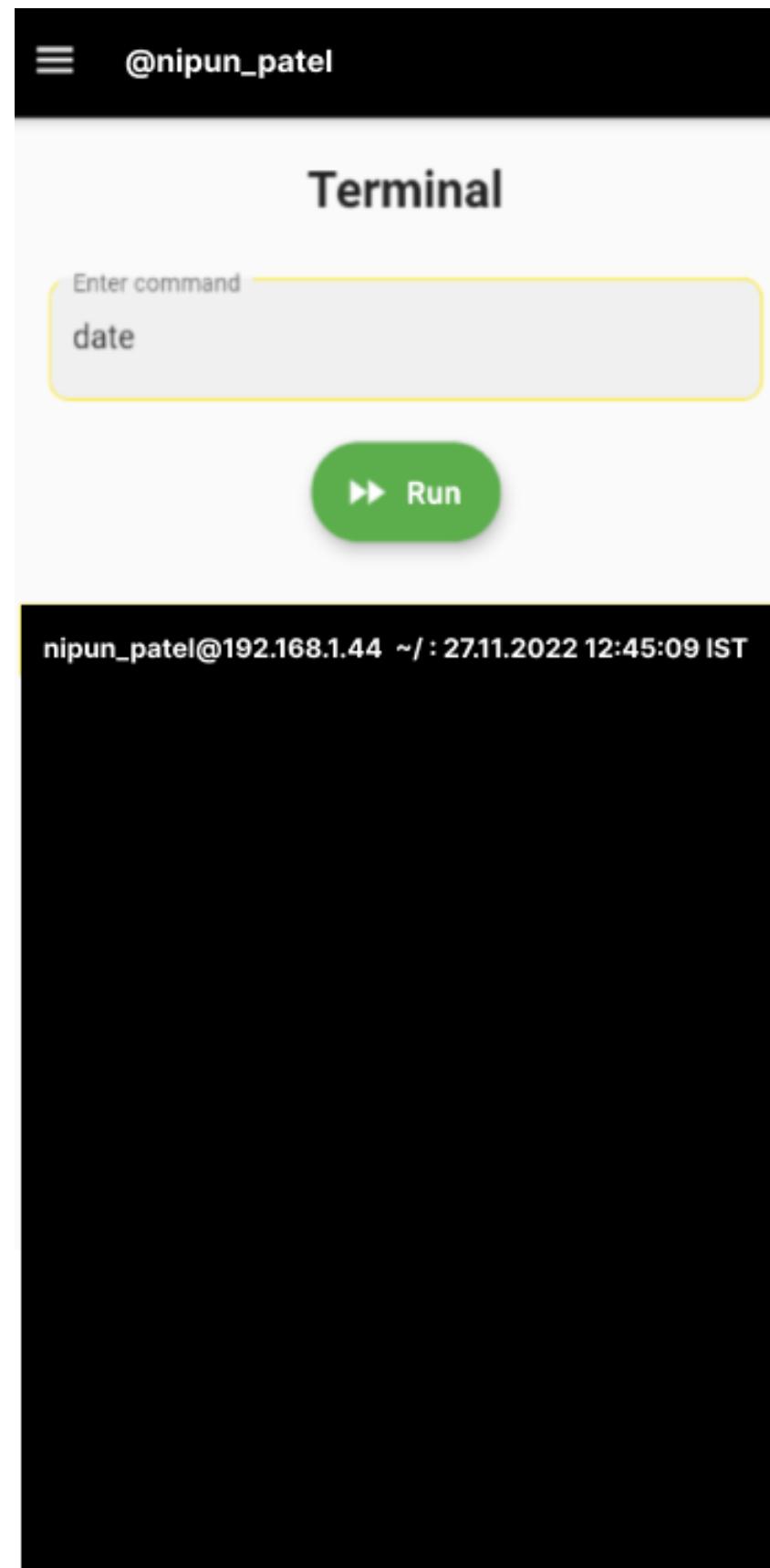
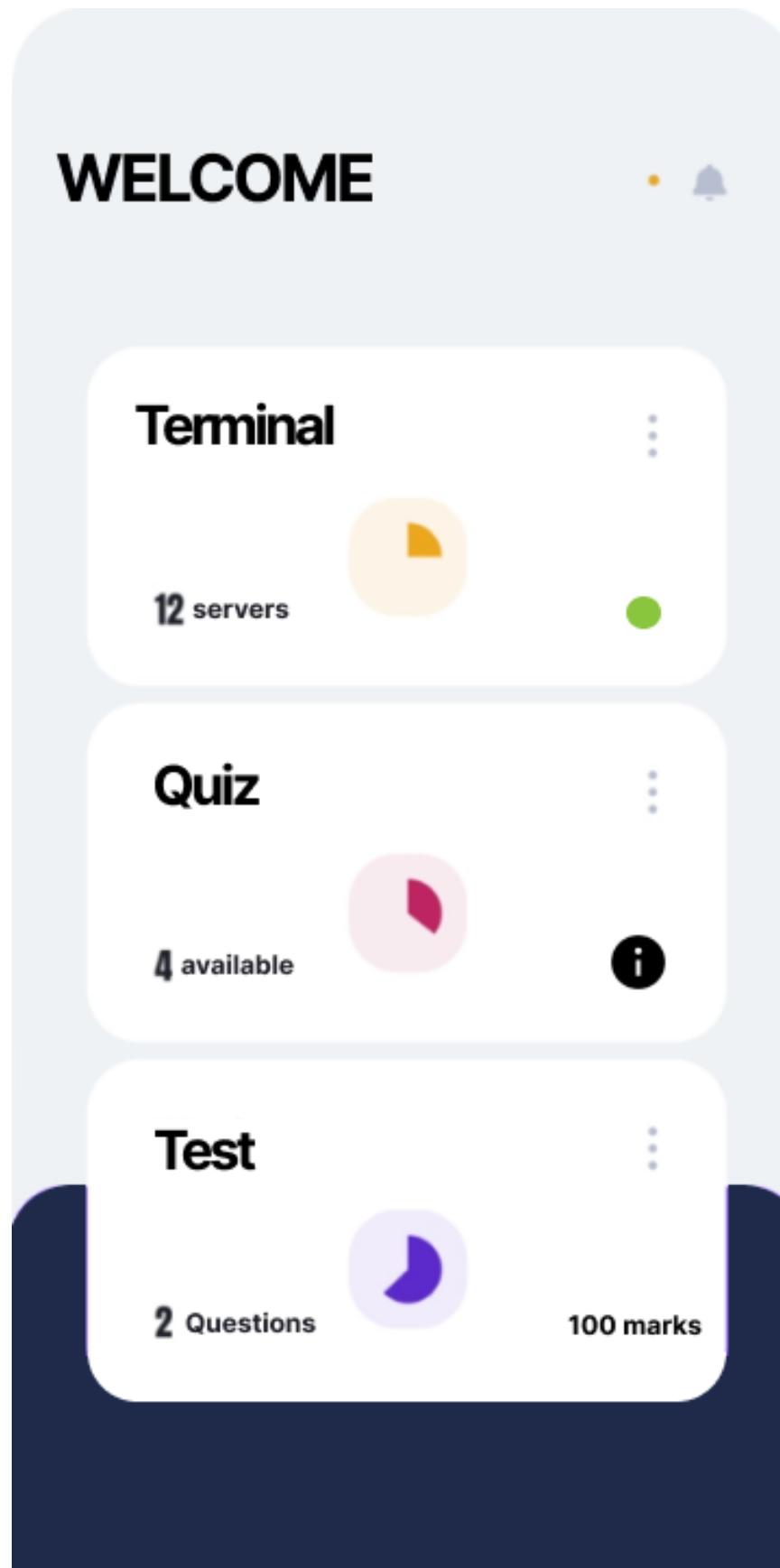
AWS cloud-hosted machine requires configuration which must be correct

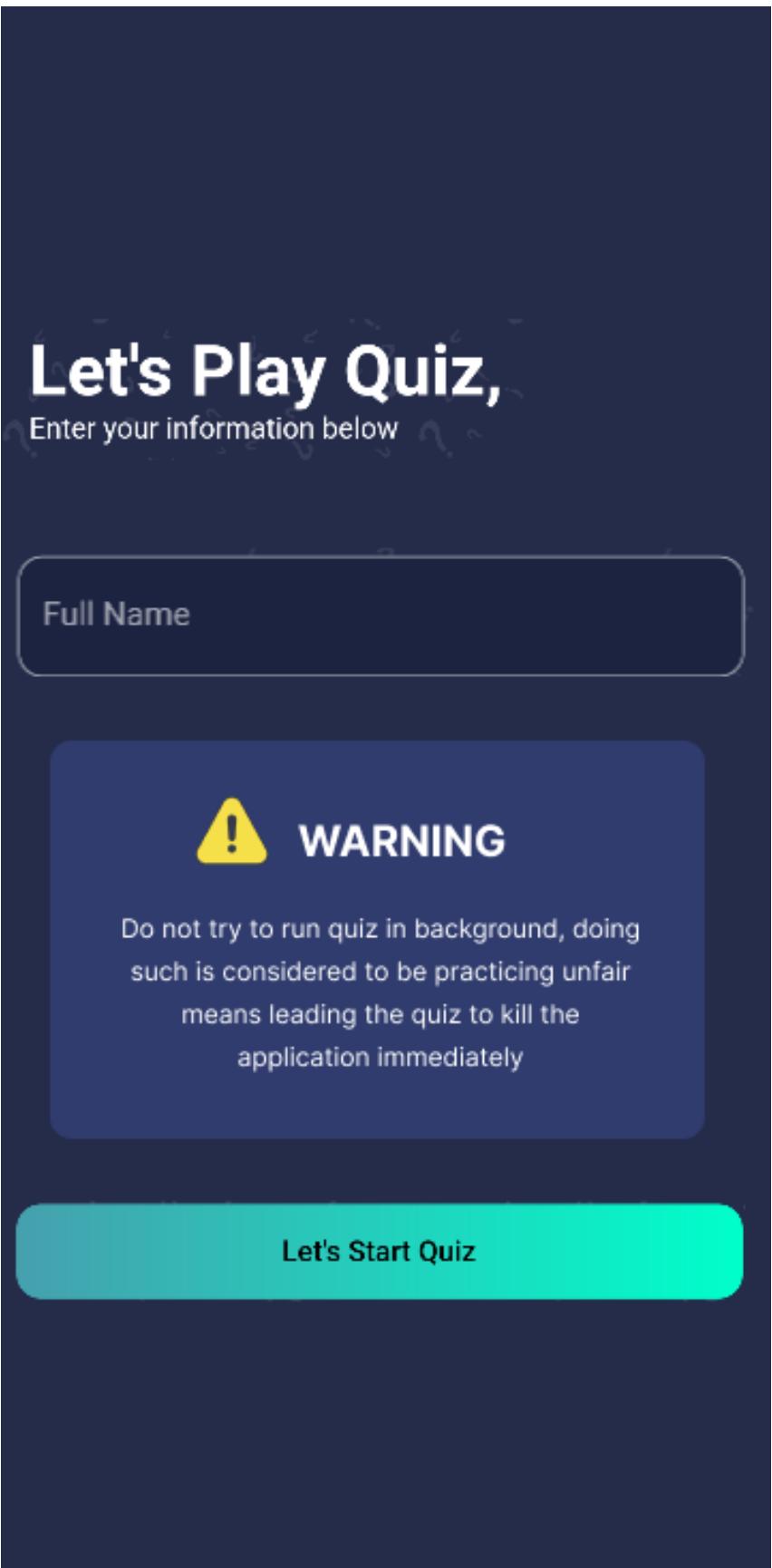




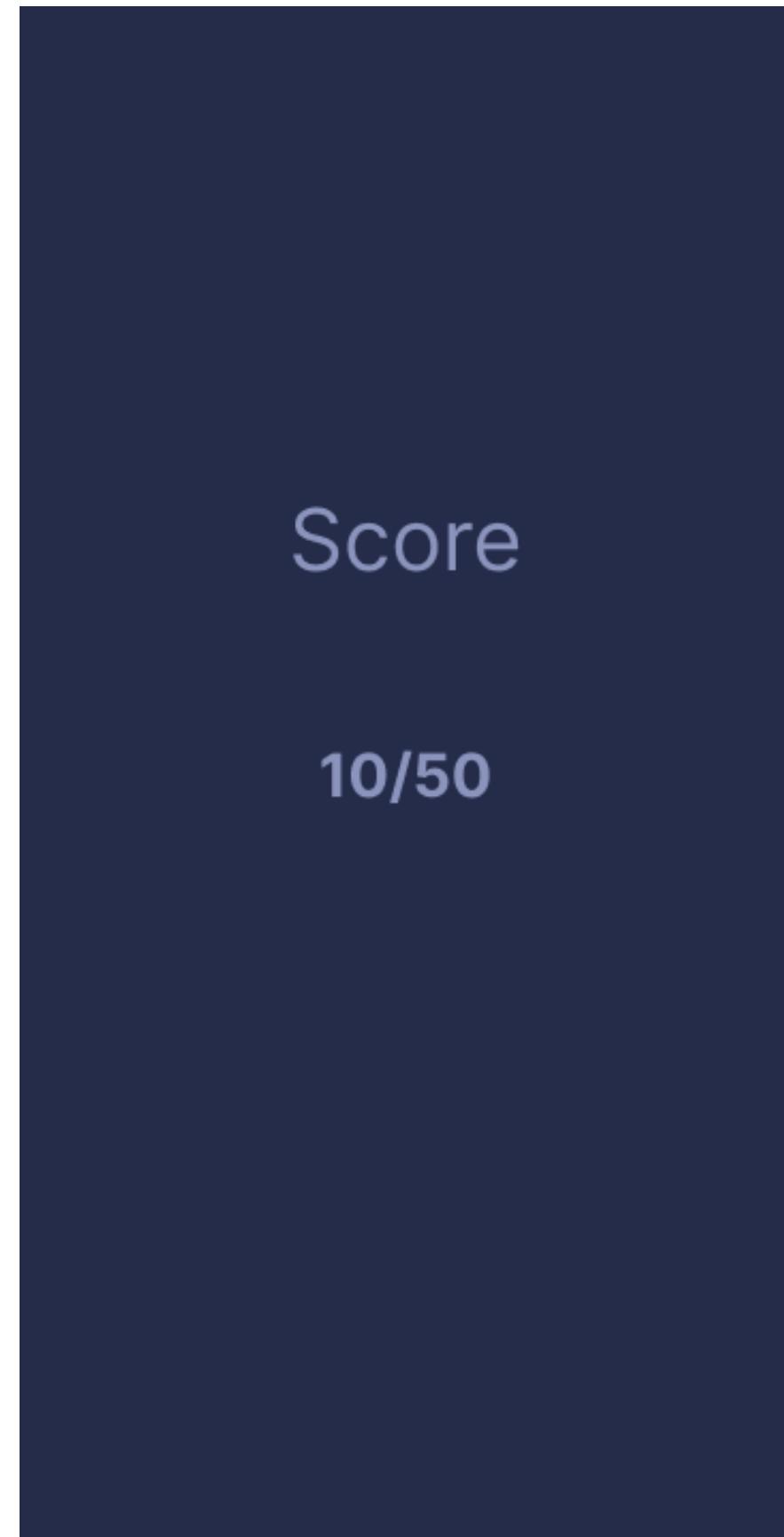
WIREFRAMES





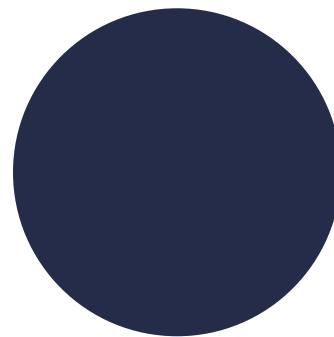


The image shows a quiz question screen. At the top, there is a timer icon showing "12 Sec" and a green checkmark icon. The text "Question 1 / 15" is displayed. The question is "Q1. Which Company has the highest no. of users_____". Below the question are four options: 1. Apple, 2. Google, 3. Facebook, and 4. Microsoft, each with a radio button next to it.

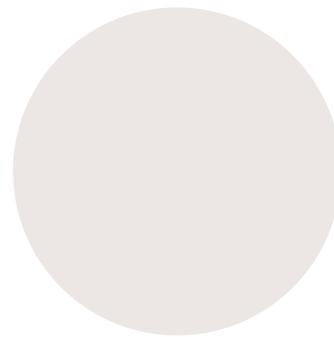




USER DEFINED STYLESHEET

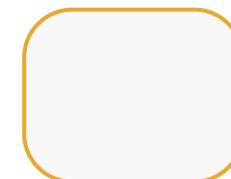
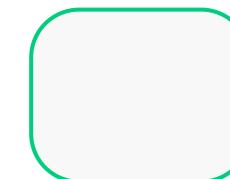
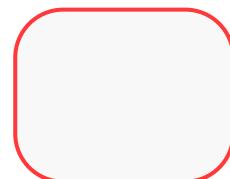


Primary Color



Secondary Color

Button rounds

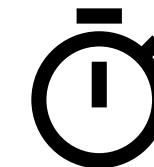


Montserrat

Inter

Inter

Icons used



Heading

Subheading

Normal text

Bold

Italic

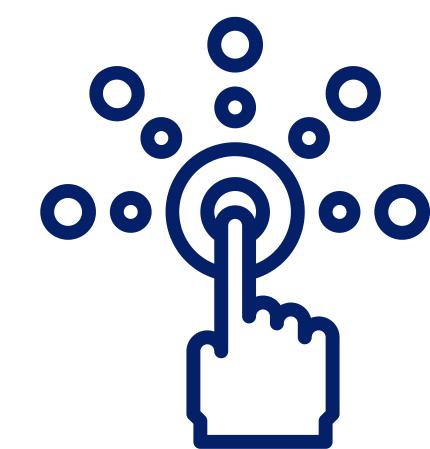
Underline



DESIGN FEATURES



Interactive Animation



Added animations for users to interact and play with the navigation features and understand app response

Consistent UI

Allow users to navigate and understand the terminal easily and use the options effectively



User defined colors

Color combination and company theme used in creating the wireframe





STAGE 4

IMPLEMENTATION



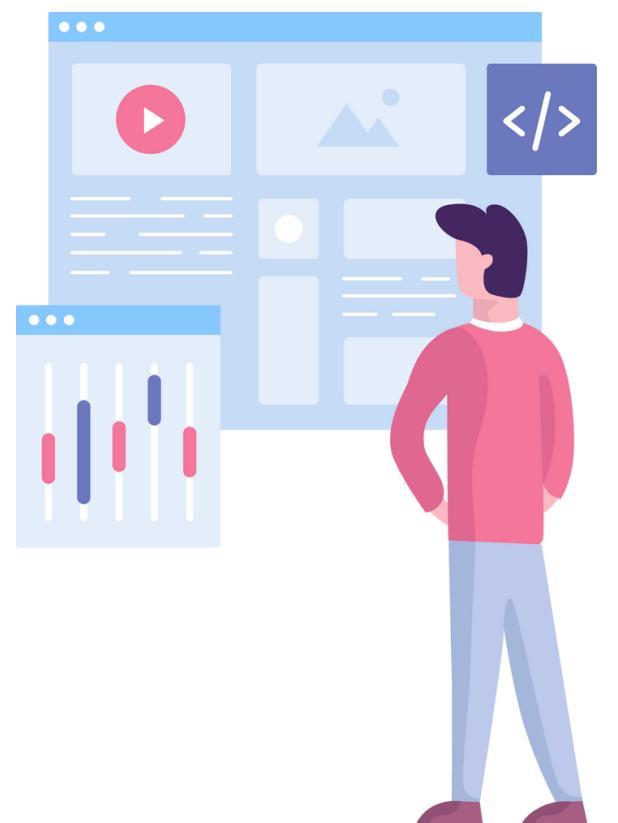
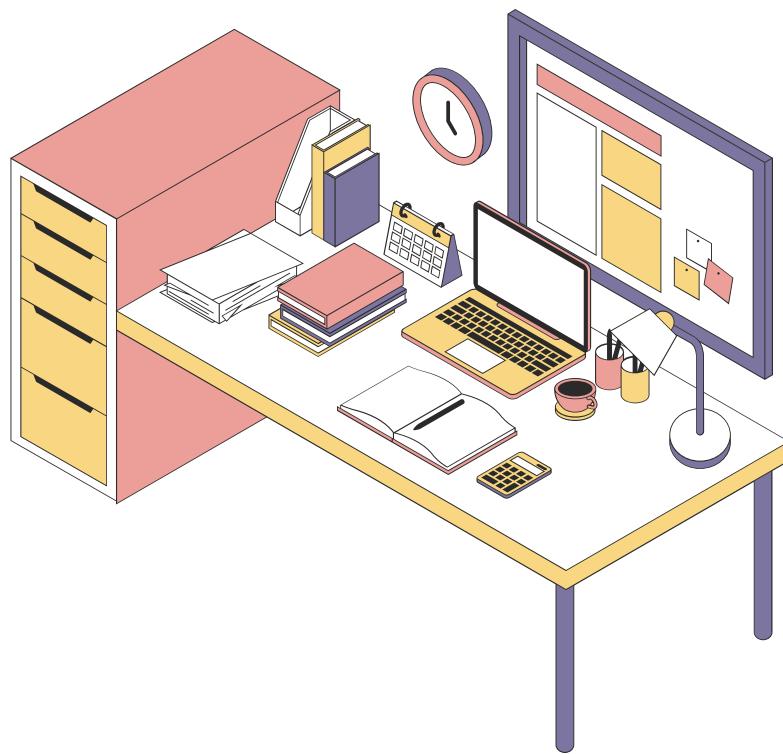
80% on progress...





STAGES

For the implementation phase, we have divided the procedure into three different stages that are setting the AWS environment, designing the backend, and connecting frontend





SETTING UP AWS ENVIRONMENT

1. Log in to the AWS console and navigate to the EC2 Instance section, and create a Redhat Linux instance free tier eligible. create a key pair credentials and save it to the desktop for login use.

The screenshot shows the AWS Lambda service interface. A modal window is open for creating a new function. In the 'Function name' field, 'travel_nipun' is entered. Under 'Runtime', 'Node.js 14.x' is selected. The 'Handler' dropdown shows 'index.handler'. The 'Memory size' is set to 128 MB, and the 'Timeout' is set to 300 seconds. The 'Billing mode' is chosen as 'Pay-as-you-go'. Below these settings, there's a 'Code' section with a 'Create new file' button and a 'Upload zip file' button. At the bottom of the modal, there are 'Cancel' and 'Create function' buttons.

The screenshot shows the AWS EC2 instance creation wizard. The first step, 'Select instance type', is completed. The 't2.micro' instance type is selected, which is free tier eligible. The next step, 'Configure Instance Details', is shown. Under 'Key pair (login)', the 'flutter-linux' key pair is selected. Under 'Network settings', the VPC and subnet are configured. On the right side, a summary panel shows the configuration details: 1 instance, Red Hat AMI, t2.micro instance type, New security group, and 1 volume (10 GiB). At the bottom right is a 'Launch instance' button.

2. Select the key pair we have created so that the machine will log in with that security credentials, next select the security rules configured in the next step and hence launch the instance.

Screenshot of the AWS CloudFront console showing the "Edit inbound rules" page. The page displays four existing security group rules and provides options to add new rules or save changes.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0458c49df96213b12	All ICMP - IPv4	ICMP	All	Custom	0.0.0.0/0
sgr-0deff7b76d804c4e9	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0a53f32e03539dcbd	SSH	TCP	22	Custom	0.0.0.0/0
sgr-0cddfb4baec054563	HTTP	TCP	80	Custom	0.0.0.0/0

Add rule

Cancel **Preview changes** **Save rules**

3. Go to the security groups we created and edit the inbound rules and give access to **HTTP**, and **HTTPS ports** as required for triggering the **web requests**. allow **ICMP ports** as they **send commands requests**, also allow **SSH Port** so that we can log in to the Linux machine with the private key we created and enable them for **public access** on **0.0.0.0/0**

The screenshot shows the AWS EC2 Instances page. The left sidebar includes options like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-options for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes). The main content displays two EC2 instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
INDIA	i-0cc927bc03ed68372	Running	t2.micro	Initializing	No alarms	ap-south-1a	ec2-3-110-30-6.ap-south-1.compute.amazonaws.com
PUBLIC	i-011fa4c54bb27622b	Running	t2.micro	Initializing	No alarms	ap-south-1a	ec2-13-233-132.ap-south-1.compute.amazonaws.com

The detailed view for the INDIA instance shows the following configuration:

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
Instance summary Instance ID: i-0cc927bc03ed68372 (INDIA) Public IPv4 address: 3.110.30.6 open address Private IPv4 addresses: 172.31.38.138 Instance state: Running Public IPv4 DNS: ec2-3-110-30-6.ap-south-1.compute.amazonaws.com open address IPv6 address: - Hostname type: IP name: ip-172-31-38-138.ap-south-1.compute.internal Private IP DNS name (IPv4 only): ip-172-31-38-138.ap-south-1.compute.internal Answer private resource DNS name: IPv4 (A) Instance type: t2.micro VPC ID: - Elastic IP addresses: - AWS Compute Optimizer finding: -						

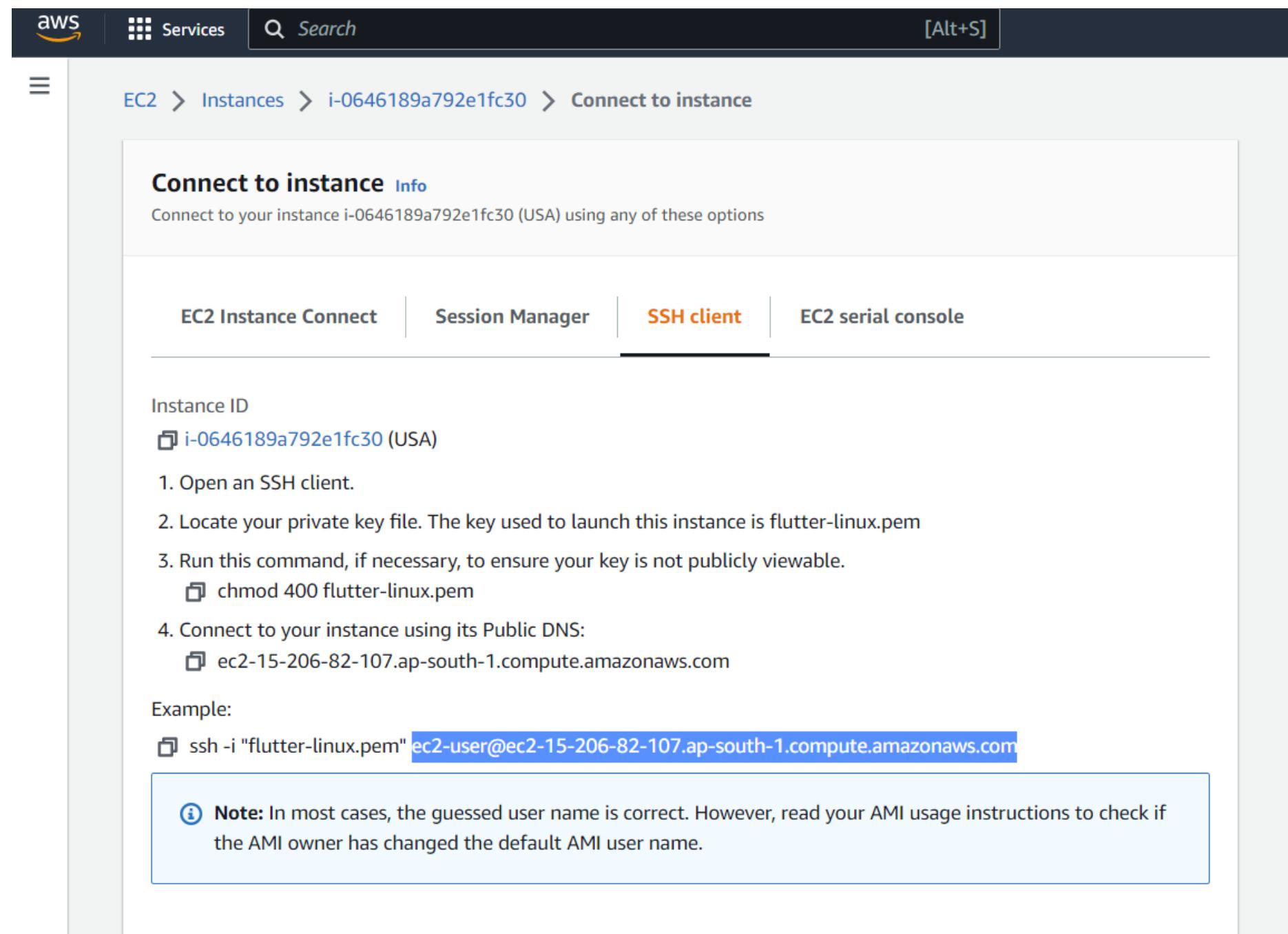
The screenshot shows the AWS EC2 Instances page. The left sidebar is identical to the first one. The main content displays one EC2 instance:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
USA	i-01f756538e93285f8	Running	t2.micro	Initializing	No alarms	us-east-1a	ec2-54-196-133-246.compute-1.amazonaws.com

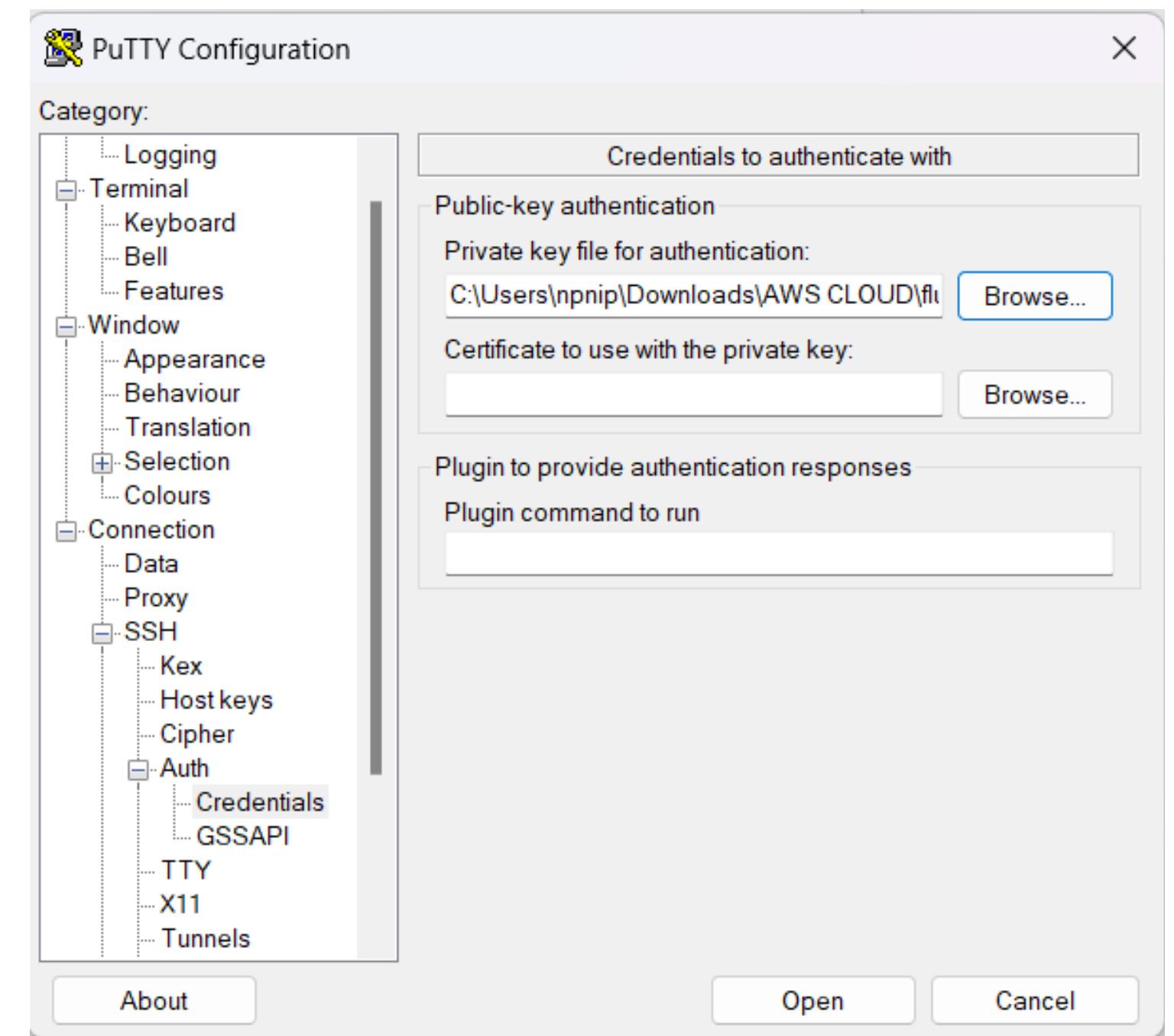
The detailed view for the USA instance shows the following configuration:

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
Instance summary Instance ID: i-01f756538e93285f8 (USA) Public IPv4 address: 54.196.133.246 open address Private IPv4 addresses: 172.31.89.32 Instance state: Running Public IPv4 DNS: ec2-54-196-133-246.compute-1.amazonaws.com open address IPv6 address: - Hostname type: IP name: ip-172-31-89-32.ec2.internal Private IP DNS name (IPv4 only): ip-172-31-89-32.ec2.internal Answer private resource DNS name: IPv4 (A) Instance type: t2.micro VPC ID: -						

4. Hence our machines are up and running we have created 3 EC2 machines available in 2 different zones **i2 in Asia (Mumbai) as ap-south-1** while others in the **USA region (N. Virginia) as us-east-1**. As a free tier eligible property we have been assigned all together a total of **750 Hrs** in one month, beyond that machines running will be charged on an hourly basis. Machines have the public IP reserved for them. if the machines will be terminated or stopped, IPs will be released back to the **reserved IP Pool** and **assigned back** if the instances run back.



The screenshot shows the AWS EC2 Instances page for an instance with ID i-0646189a792e1fc30. The 'Connect to instance' section is open, showing the 'SSH client' tab selected. It provides instructions for connecting via SSH, including steps to open an SSH client, locate the private key file (flutter-linux.pem), run chmod 400 on it, and connect using the Public DNS (ec2-15-206-82-107.ap-south-1.compute.amazonaws.com). An example command is provided: ssh -i "flutter-linux.pem" ec2-user@ec2-15-206-82-107.ap-south-1.compute.amazonaws.com. A note at the bottom states: "Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name."



The screenshot shows the PuTTY Configuration dialog. The 'Category' tree on the left includes Terminal, Window, Connection, and SSH sections. Under 'Connection', the 'Auth' section is expanded, showing 'Credentials' and 'GSSAPI' options. On the right, there are fields for 'Credentials to authenticate with' (Public-key authentication), 'Private key file for authentication' (C:\Users\npnip\Downloads\AWS CLOUD\flutter-linux.pem, with a 'Browse...' button), and 'Certificate to use with the private key' (with a 'Browse...' button). Below these are sections for 'Plugin to provide authentication responses' and 'Plugin command to run'.

5. Now its time to configure our Linux server for that we need to connect to the machine using putty client, copy the ssh client URL and paste it the putty, now add the security key we have downloaded for putty it will come under **SSH---->AUTH---->Credentials** part, browse your key add it and connect it.

2

SERVER AND BACKEND CONFIGURATION

```
[root@ip-172-31-32-22 ~]# yum install httpd*
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use subscription-manager to register.

Red Hat Enterprise Linux 9 for x86_64 - AppStream 38 MB/s | 15 MB 00:00
```

→

```
[root@ip-172-31-32-22 ~]# yum install httpd*
perl-overloading noarch 0.02-479.el9 rhel-9-appstream-rhui-rpms 23 k
perl-parent noarch 1:0.238-460.el9 rhel-9-appstream-rhui-rpms 16 k
perl-podlators noarch 1:4.14-460.el9 rhel-9-appstream-rhui-rpms 118 k
perl-subs noarch 1.03-479.el9 rhel-9-appstream-rhui-rpms 22 k
perl-vars noarch 1.05-479.el9 rhel-9-appstream-rhui-rpms 23 k
pkgconf x86_64 1.7.3-9.el9 rhel-9-baseos-rhui-rpms 45 k
pkgconf-m4 noarch 1.7.3-9.el9 rhel-9-baseos-rhui-rpms 16 k
pkgconf-pkg-config x86_64 1.7.3-9.el9 rhel-9-baseos-rhui-rpms 12 k
redhat-logos-httd noarch 90.4-1.el9 rhel-9-appstream-rhui-rpms 18 k
Installing weak dependencies:
apr-util-openssl x86_64 1.6.1-20.el9 rhel-9-appstream-rhui-rpms 17 k
mod_http2 x86_64 1.15.19-2.el9 rhel-9-appstream-rhui-rpms 153 k
mod_lua x86_64 2.4.53-7.el9 rhel-9-appstream-rhui-rpms 64 k
perl-IO-Socket-SSL noarch 2.073-1.el9 rhel-9-appstream-rhui-rpms 223 k
perl-Mozilla-CA noarch 20200520-6.el9 rhel-9-appstream-rhui-rpms 14 k
perl-NDBM_File x86_64 1.15-479.el9 rhel-9-appstream-rhui-rpms 33 k
Transaction Summary
=====
Install 82 Packages

Total download size: 13 M
Installed size: 45 M
Is this ok [y/N]: y
```

→

```
[root@ip-172-31-32-22 ~]# yum install httpd*
perl-Text-ParseWords-3.30-460.el9.noarch
perl-Text-Tabs+Wrap-2013.0523-460.el9.noarch
perl-Time-Local-2:1.300-7.el9.noarch
perl-URI-5.09-3.el9.noarch
perl-base-2.27-479.el9.noarch
perl-constant-1.33-461.el9.noarch
perl-if-0.60.800-479.el9.noarch
perl-interpreter-4:5.32.1-479.el9.x86_64
perl-libnet-3.13-4.el9.noarch
perl-libs-4:5.32.1-479.el9.x86_64
perl-mro-1.23-479.el9.x86_64
perl-overload-1.31-479.el9.noarch
perl-overloading-0.02-479.el9.noarch
perl-parent-1:0.238-460.el9.noarch
perl-podlators-1:4.14-460.el9.noarch
perl-subs-1.03-479.el9.noarch
perl-vars-1.05-479.el9.noarch
pkgconf-1.7.3-9.el9.x86_64
pkgconf-m4-1.7.3-9.el9.noarch
pkgconf-pkg-config-1.7.3-9.el9.x86_64
redhat-logos-httd-90.4-1.el9.noarch

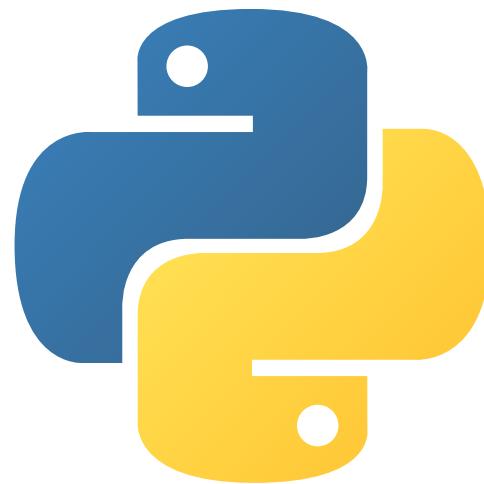
Complete!
[root@ip-172-31-32-22 ~]# yum install python3*
```

```
[root@ip-172-31-32-22 ~]# yum install python3*
libproxy-webkitgtk4 x86_64 0.4.15-35.el9 rhel-9-appstream-rhui-rpms 22 k
libvirt-daemon-config-network
x86_64 8.5.0-7.el9_1 rhel-9-appstream-rhui-rpms 21 k
mesa-dri-drivers x86_64 22.1.5-2.el9 rhel-9-appstream-rhui-rpms 9.2 M
nmap-ncat x86_64 3:7.91-10.el9 rhel-9-appstream-rhui-rpms 230 k
p11-kit-server x86_64 0.24.1-2.el9 rhel-9-appstream-rhui-rpms 200 k
perl-Devel-Peek x86_64 1.28-479.el9 rhel-9-appstream-rhui-rpms 44 k
pipewire x86_64 0.3.47-2.el9_0 rhel-9-appstream-rhui-rpms 41 k
pipewire-alsa x86_64 0.3.47-2.el9_0 rhel-9-appstream-rhui-rpms 61 k
pipewire-jack-audio-connection-kit
x86_64 0.3.47-2.el9_0 rhel-9-appstream-rhui-rpms 135 k
pipewire-pulseaudio x86_64 0.3.47-2.el9_0 rhel-9-appstream-rhui-rpms 26 k
tracker-miners x86_64 3.1.2-1.el9 rhel-9-appstream-rhui-rpms 944 k
xdg-desktop-portal-gtk
x86_64 1.12.0-3.el9 rhel-9-appstream-rhui-rpms 139 k
xdg-utils noarch 1.1.3-11.el9 rhel-9-appstream-rhui-rpms 78 k
Transaction Summary
=====
Install 647 Packages
Upgrade 64 Packages

Total download size: 512 M
Is this ok [y/N]: y
```



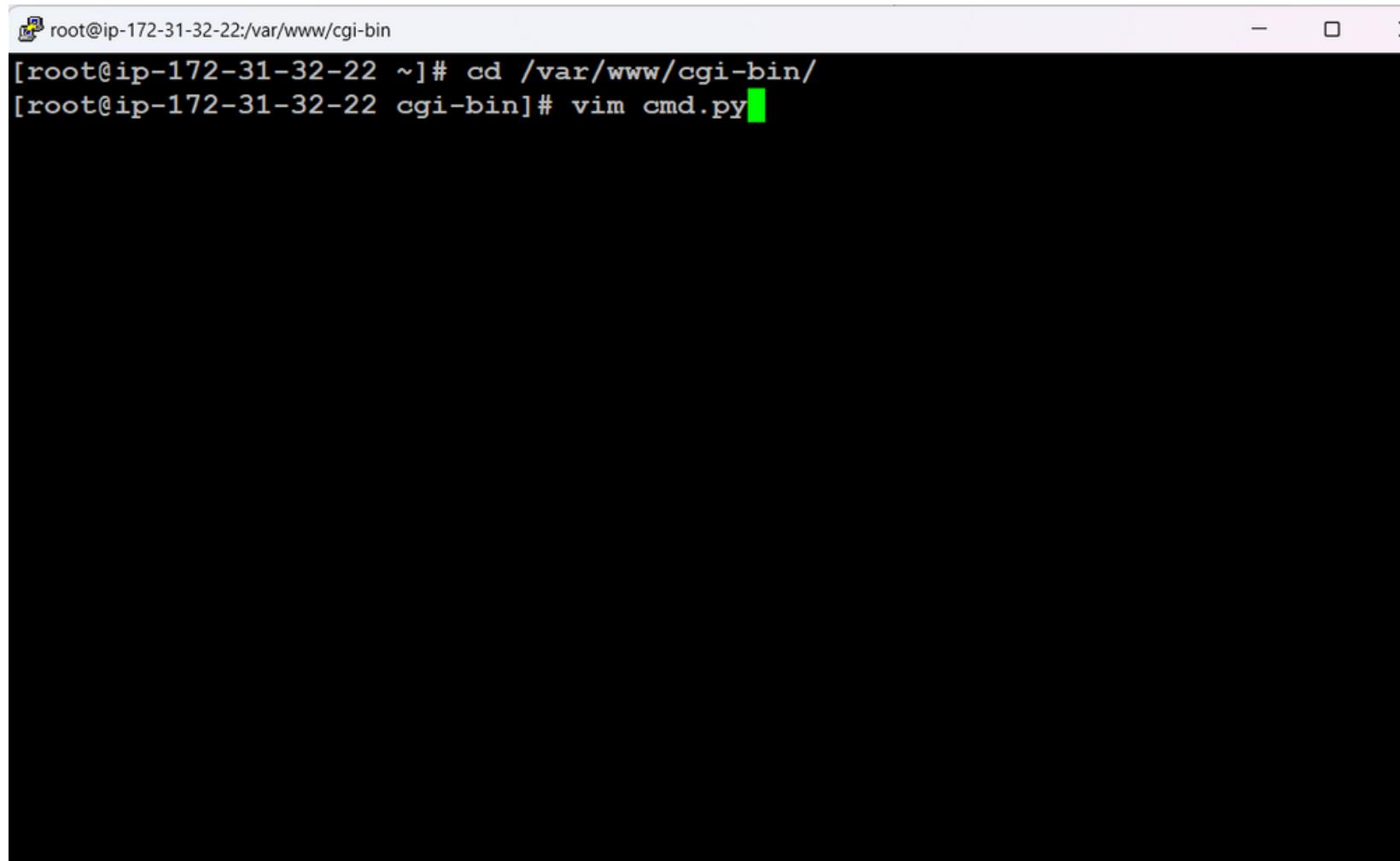
Apache



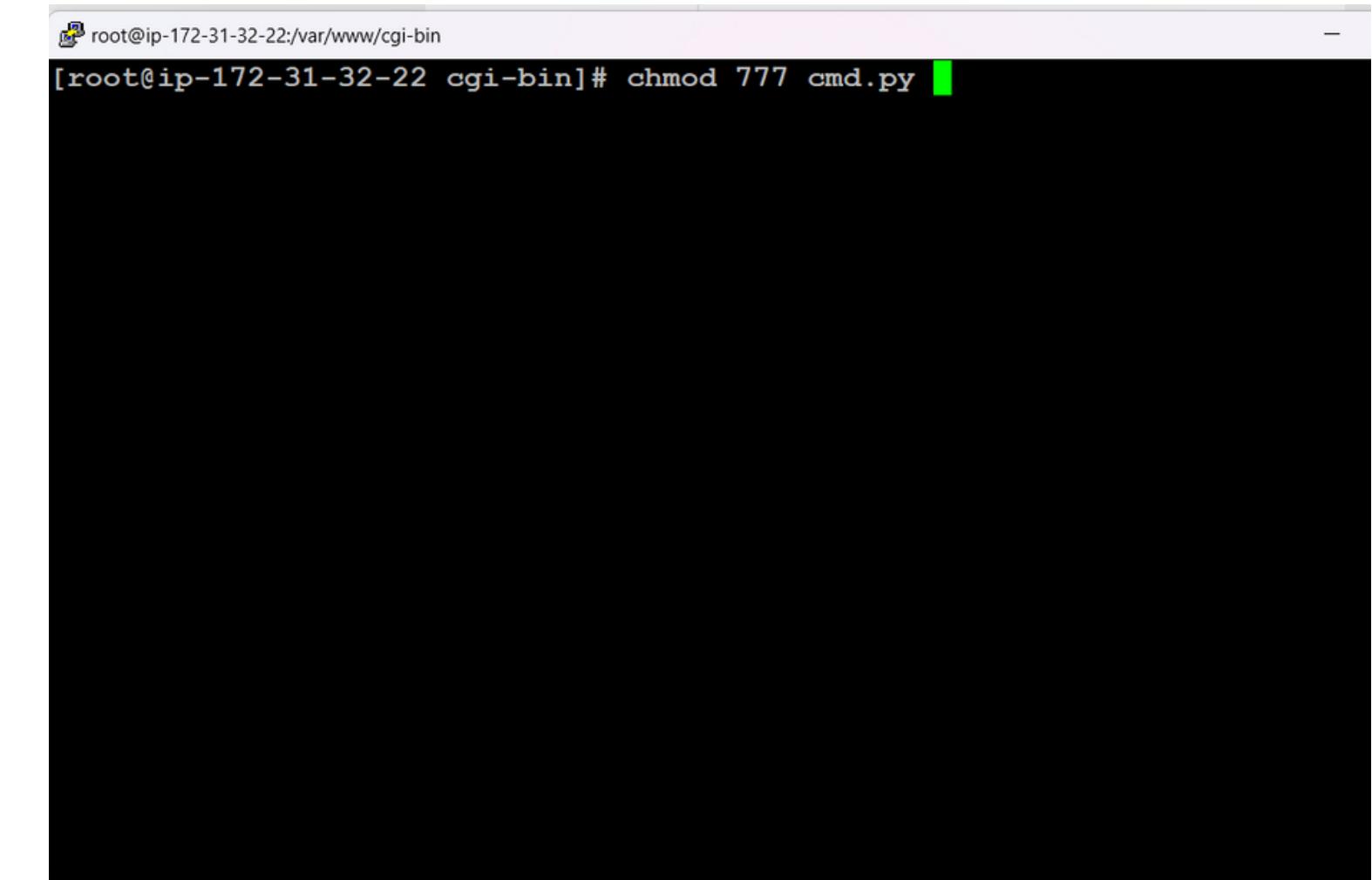
Python

1. As we are using the web requests we need to configure the apache server as it is responsible for hosting the web resources therefore it can be configured using the YUM server the command initiated is **yum install httpd***

while we also installed the python service as it will be the communicating host in understanding the language of commands and executing the same via webhooks received the command for the same is **yum install python3**



```
root@ip-172-31-32-22:/var/www/cgi-bin
[root@ip-172-31-32-22 ~]# cd /var/www/cgi-bin/
[root@ip-172-31-32-22 cgi-bin]# vim cmd.py
```



```
root@ip-172-31-32-22:/var/www/cgi-bin
[root@ip-172-31-32-22 cgi-bin]# chmod 777 cmd.py
```

2. Now we will be creating the python script in the cgi-bin as apache files are handled in that folder only. The command of execution is **cd /var/www/cgi-bin/**

Inside this we will create a python file of any user-based choice say **cmd.py** and give this file user, owner, and group permission as required for execution, the command for the same is **chmod 777 cmd.py**, where **7** denotes **RWX-- permissions** for the respective domain as discussed

```
root@ip-172-31-32-22:/var/www/cgi-bin
#!/usr/bin/python3

import cgi
import subprocess
import json

print("content-type: text/html")
print()

user_input = cgi.FieldStorage()
command = user_input.getvalue("x")
output = subprocess.getstatusoutput("{0}".format(command))
StatusCode = output[0]
result = output[1]
db = { "Result": result, "statusCode": StatusCode }
finalResult = json.dumps(db)
print(finalResult)

-- INSERT --
```

3. This is the python scripting we have done, we have imported the common gateway interface and it will only work if we have the required dependencies installed on the machine such as python3, explanation of code is presented at right side

cgi.FieldStorage-----> this will create the instance for the input given

command----->this will take user input

subprocess.getstatusoutput-----> this will process the command and return 0 as the status code if the command given is correct else will give as 1

correct and incorrect status are stored in separate variables as **StatusCode** and **result**

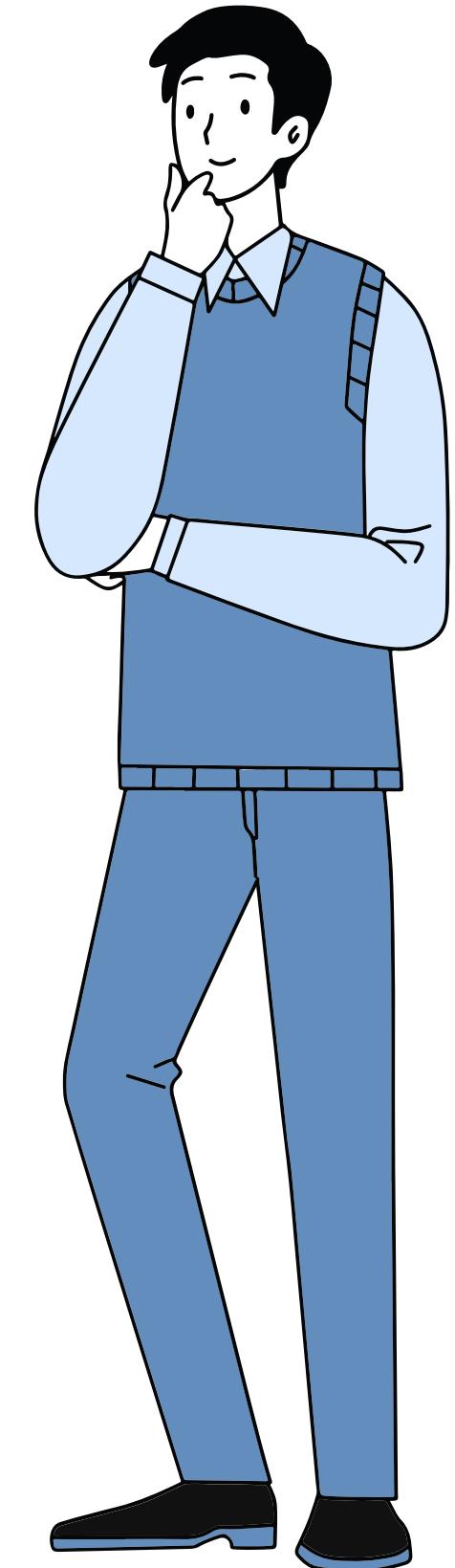
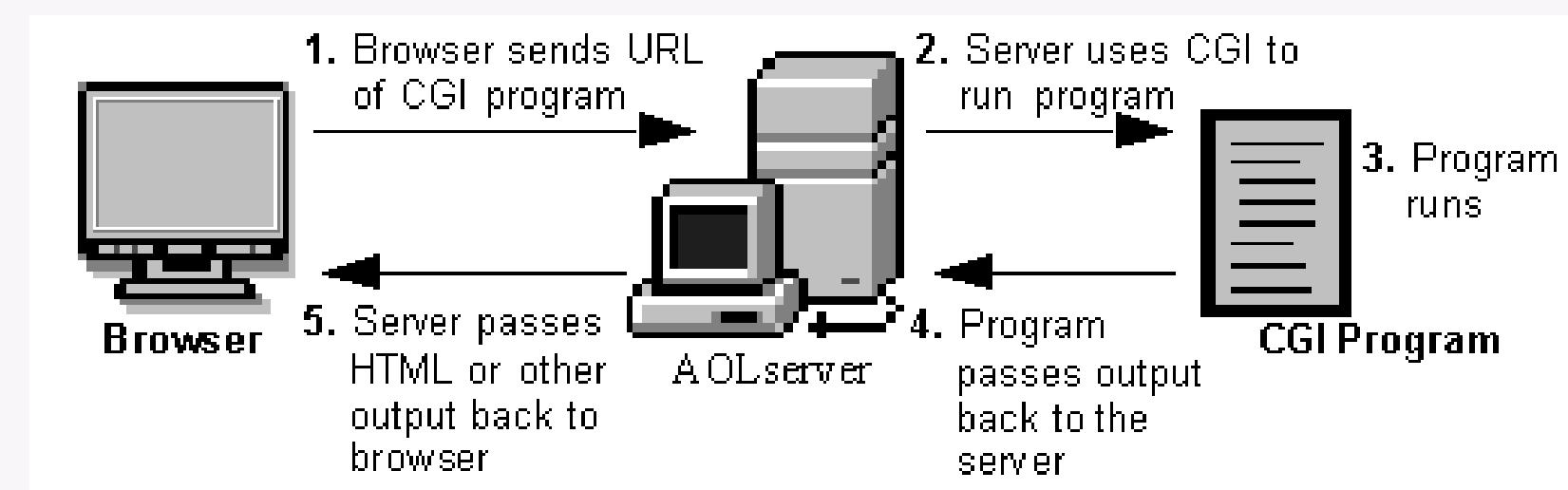
db-----> will create a JSON format and the final result will serialize the same and send back to the host requesting it.

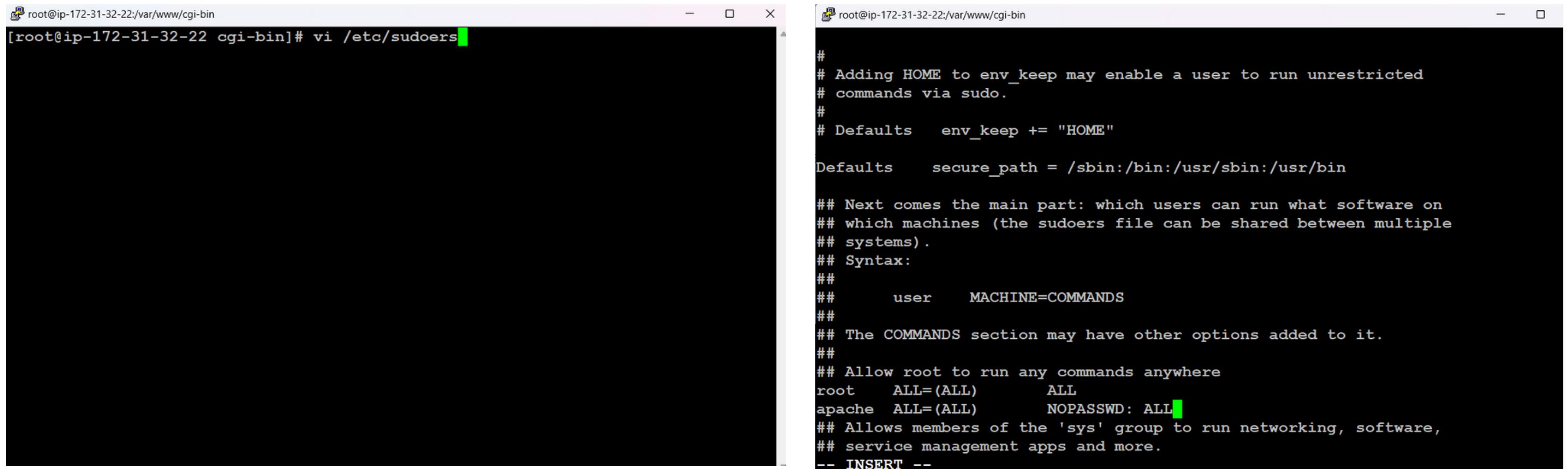
What is CGI(Common Gateway Interface) and how does it works.....?

CGI (Common Gateway Interface) is a standard way of running programs from a Web server. Often, CGI programs are used to generate pages dynamically or to perform some other action when someone fills out an HTML form and clicks the submit button.

Basically, CGI works like this:

A **reader sends a URL** that causes the **server** to use **CGI to run a program**. The server passes input from the reader to the program and output from the program back to the reader. CGI acts as a "gateway" between the server and the program you write. CGI is a standard interface used by many **Web servers**, there are lots of example programs and function libraries available on the **World Wide Web and by FTP**.





The image shows two terminal windows side-by-side. Both windows have a title bar with the text "root@ip-172-31-32-22:/var/www/cgi-bin". The left window shows the command "[root@ip-172-31-32-22 cgi-bin]# vi /etc/sudoers" followed by a large black redacted area. The right window shows the contents of the /etc/sudoers file:

```
#  
# Adding HOME to env_keep may enable a user to run unrestricted  
# commands via sudo.  
#  
# Defaults env_keep += "HOME"  
  
Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin  
  
## Next comes the main part: which users can run what software on  
## which machines (the sudoers file can be shared between multiple  
## systems).  
## Syntax:  
##  
##       user      MACHINE=COMMANDS  
##  
## The COMMANDS section may have other options added to it.  
##  
## Allow root to run any commands anywhere  
root    ALL=(ALL)      ALL  
apache  ALL=(ALL)      NOPASSWD: ALL  
## Allows members of the 'sys' group to run networking, software,  
## service management apps and more.  
-- INSERT --
```

4. Now we need to provide root permissions to the apache server as if we need to provide user-based access to the terminal, not the root user as for security reasons, the command of execution will be [vi /etc/sudoers](#)

add the line below the root permission as **apache ALL=(ALL) NOPASSWD: ALL**

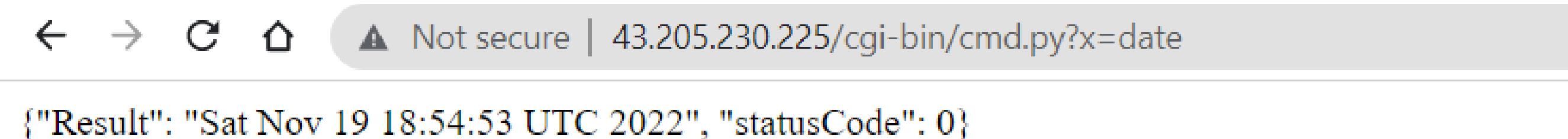
The above line denotes to give all root access to apache while NOPASSWD gives permission to access by any user without any password

```
[root@ip-172-31-32-22 cgi-bin]# vi /etc/sudoers  
[root@ip-172-31-32-22 cgi-bin]# systemctl enable httpd.service  
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /u  
/lib/systemd/system/httpd.service.  
[root@ip-172-31-32-22 cgi-bin]#
```

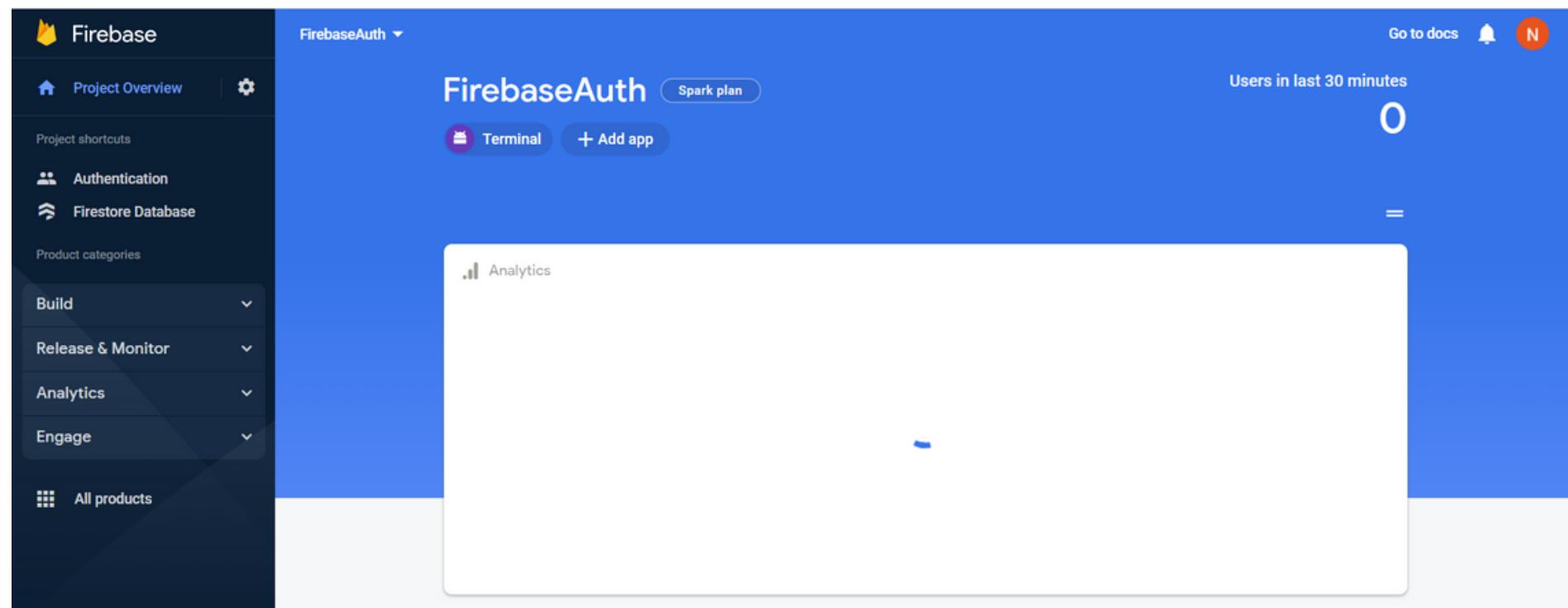
```
[root@ip-172-31-32-22 cgi-bin]# vi /etc/sudoers  
[root@ip-172-31-32-22 cgi-bin]# systemctl enable httpd.service  
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /u  
/lib/systemd/system/httpd.service.  
[root@ip-172-31-32-22 cgi-bin]# systemctl start httpd.service  
[root@ip-172-31-32-22 cgi-bin]#
```

5. Now enable and start the apache server, the command for the same is systemctl enable httpd.service and systemctl start httpd

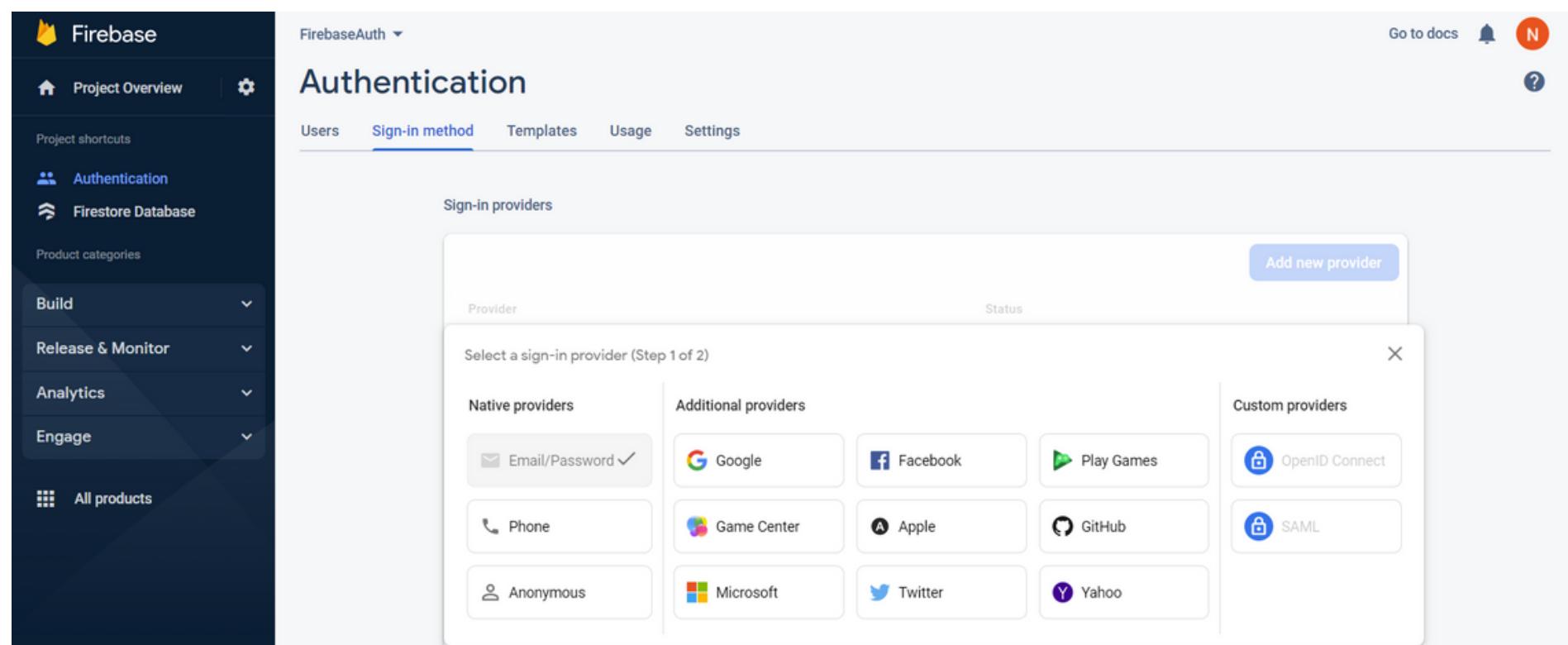
Now its time to check whether our configurations are working perfectly or not, so we will **go to the browser** and **add** our **public IP** and **location of our python file** with the command we want to execute, for testing we used the **date** as a command,



Hence we can see our command is being executed and we can see the JSON result with status code as 0 on successful command execution, now we need to prepare our backend to store the history of execution for each user of IANT



6. Now for the backend in storing the data we need a service that is firebase responsible for handling data requests. **create a firebase project and add authentication and firestore database services to your project.**

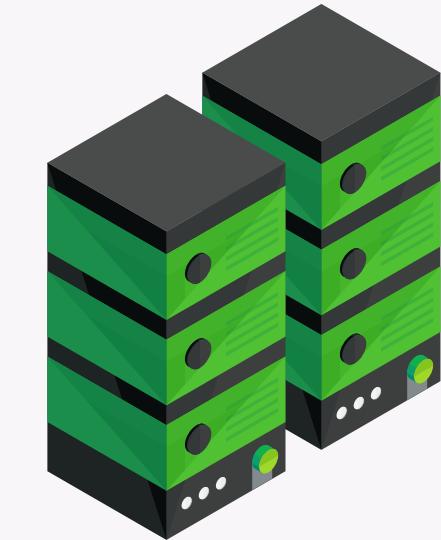
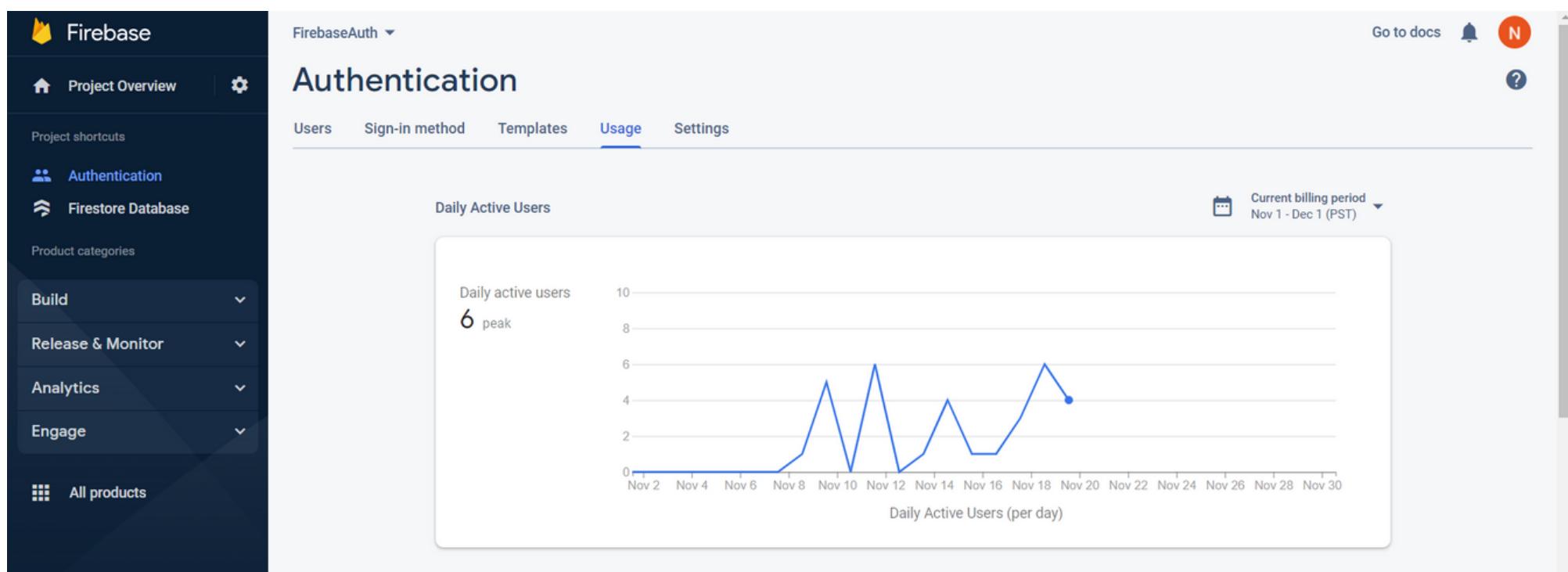


Now register your mobile application on this firebase project by checking your package name found **android----->app----->src----->main----->manifestfile** add the package name.

Now **enable the authentication** required for your user in our case we are providing **email authentication**.

The screenshot shows the Firebase Authentication console. The left sidebar has 'Authentication' selected under 'Project shortcuts'. The main area displays a table of users with columns: Identifier, Providers, Created, Signed In, and User UID. A single user is listed: Identifier 'a@a.com', Providers 'Email', Created 'Nov 20, 2022', Signed In 'Nov 20, 2022', and User UID '7dg9bBq6I7bV6PUdmuaEb9tYhPI3'. There are buttons for 'Add user' and a three-dot menu.

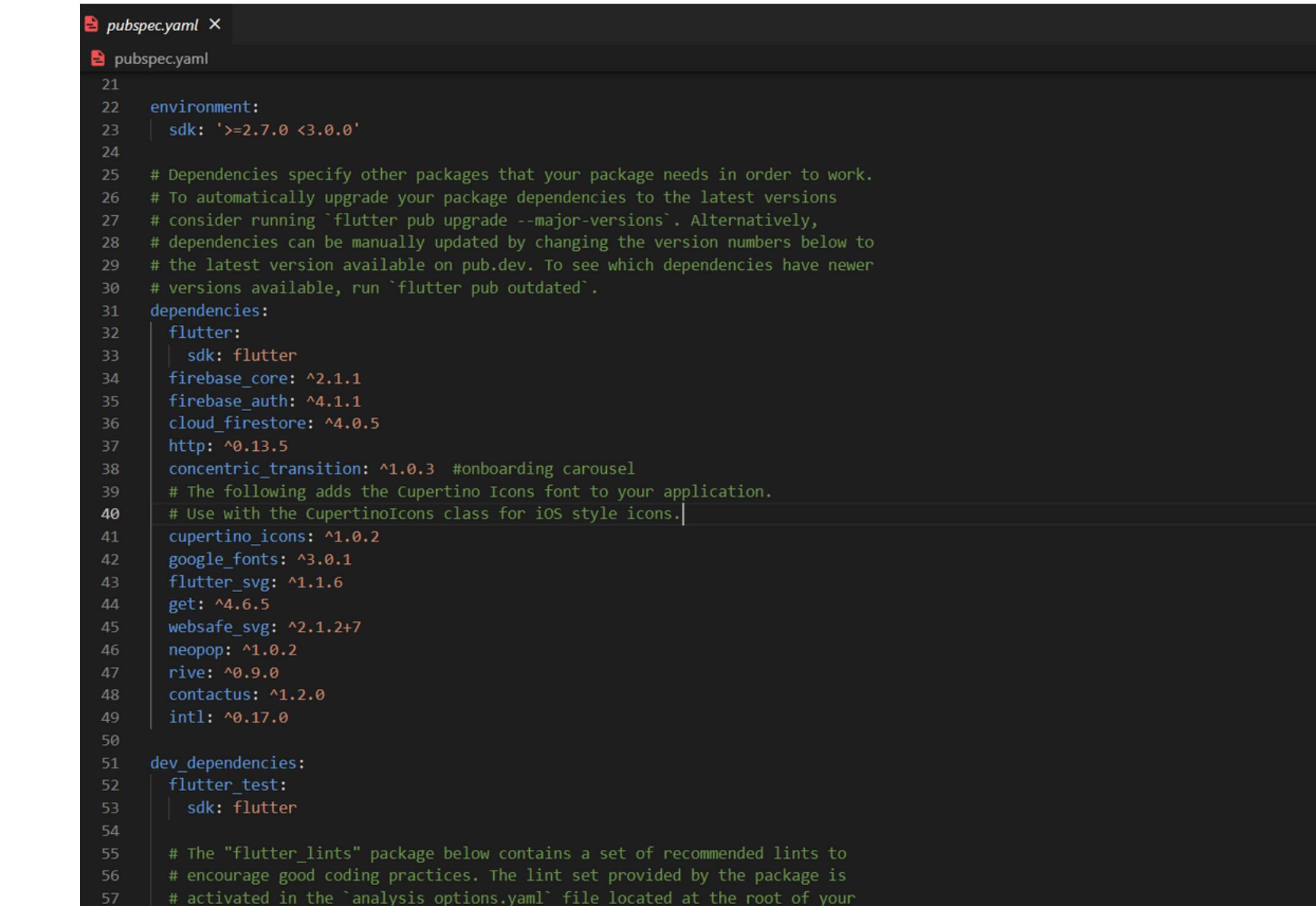
The screenshot shows the Cloud Firestore console. The left sidebar has 'Cloud Firestore' selected under 'Project shortcuts'. The main area shows a document structure under 'Users': 'fir-auth-9d469' -> 'Users' -> '7dg9bBq6I7bV6'. It includes options to 'Start collection', 'Add document', and 'Add field'. Below the document tree, there are fields for 'Result' and 'statusCode'.



7. Monitor your user activity by checking the users in **authentication**, you own the whole right to **delete any user** without letting them, all their **data will be destroyed**, also in our database we can see the individual users' command data with the **result and status code** provided, and we can monitor our users log in and access our service and no. of times the user logged into the database. **Firebase has this limitation, it allows 10k users to sign in per month in total beyond that the service is chargeable.**

3

FRONTEND, BACKEND CONNECTION

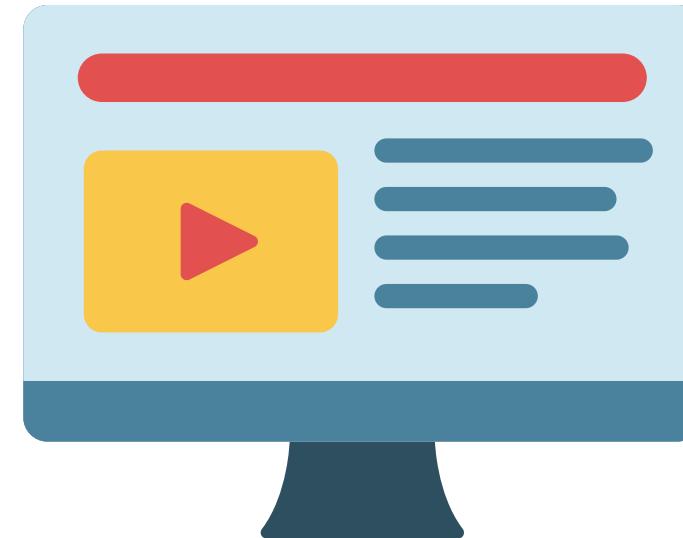


```
pubspec.yaml
environment:
  sdk: '>=2.7.0 <3.0.0'

# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^2.1.1
  firebase_auth: ^4.1.1
  cloud_firestore: ^4.0.5
  http: ^0.13.5
  concentric_transition: ^1.0.3 #onboarding carousel
  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  google_fonts: ^3.0.1
  flutter_svg: ^1.1.6
  get: ^4.6.5
  websafe_svg: ^2.1.2+7
  neopop: ^1.0.2
  rive: ^0.9.0
  contactus: ^1.2.0
  intl: ^0.17.0

dev_dependencies:
  flutter_test:
    sdk: flutter

# The "flutter_lints" package below contains a set of recommended lints to
# encourage good coding practices. The lint set provided by the package is
# activated in the `analysis_options.yaml` file located at the root of your
```



1. Add the respective dependencies in **pubsec.yaml** file which is basically the registry of packages we are using in our project. Add the **firebase, authentication** and **other dependencies** whichever is required.



```
choice_page.dart ×
lib > pages > choice_page.dart > choice > build
50   Container(
51     width: double.maxFinite,
52     height: 400,
53     child: RiveAnimation.asset(
54       "assets/this that.riv",
55       fit: BoxFit.fill,
56     ), // RiveAnimation.asset
57   ), // Container
58   NeoPopTiltedButton(
59     isFloating: true,
60     onTapUp: () {
61       Navigator.of(context).push(
62         MaterialPageRoute(
63           builder: (context) => LoginPage(),
64         ), // MaterialPageRoute
65       );
66     },
67     decoration: const NeoPopTiltedButtonDecoration(
68       color: Color.fromARGB(255, 22, 72, 137),
69       plunkColor: Color.fromARGB(255, 22, 72, 137),
70       shadowColor: Color.fromRGBO(36, 36, 36, 1),
71       showShimmer: true,
72     ), // NeoPopTiltedButtonDecoration
73     child: const Padding(
74       padding: EdgeInsets.symmetric(
75         horizontal: 70.0,
76         vertical: 15,
77       ), // EdgeInsets.symmetric
78     ),
79     child: Text(
80       'IANT USER',
81       style: TextStyle(
82         color: Colors.white,
83         fontWeight: FontWeight.bold,
84         fontSize: 15,
85       ), // TextStyle
86     ), // Text
87   ), // Padding
```

```
choice_page.dart ×
lib > pages > choice_page.dart > choice > build
5 > import 'package:terminal/pages/dashboard/screens/product/public_terminal.dart';
6 import 'package:terminal/pages/dashboard/screens/product/terminal_page.dart';
7 import 'package:terminal/pages/login_register_page.dart';
8
9 class choice extends StatelessWidget {
10   Widget build(BuildContext context) {
11     return Scaffold(
12       backgroundColor: Color(0xff2D5F7B),
13       body: Container(
14         width: double.infinity,
15         height: double.infinity,
16         child: Column(
17           mainAxisAlignment: MainAxisAlignment.center,
18           mainAxisSize: MainAxisSize.center,
19           children: [
20             NeoPopTiltedButton(
21               isFloating: true,
22               onTapUp: () {
23                 Navigator.of(context).push(
24                   MaterialPageRoute(
25                     builder: (context) => public_cli(),
26                   ), // MaterialPageRoute
27                 );
28               },
29               decoration: const NeoPopTiltedButtonDecoration(
30                 color: Color.fromARGB(255, 22, 72, 137),
31                 plunkColor: Color.fromARGB(255, 22, 72, 137),
32                 shadowColor: Color.fromRGBO(36, 36, 36, 1),
33               ), // NeoPopTiltedButtonDecoration
34             ),
35           ],
36         ),
37       ),
38     );
39   }
40 }
```

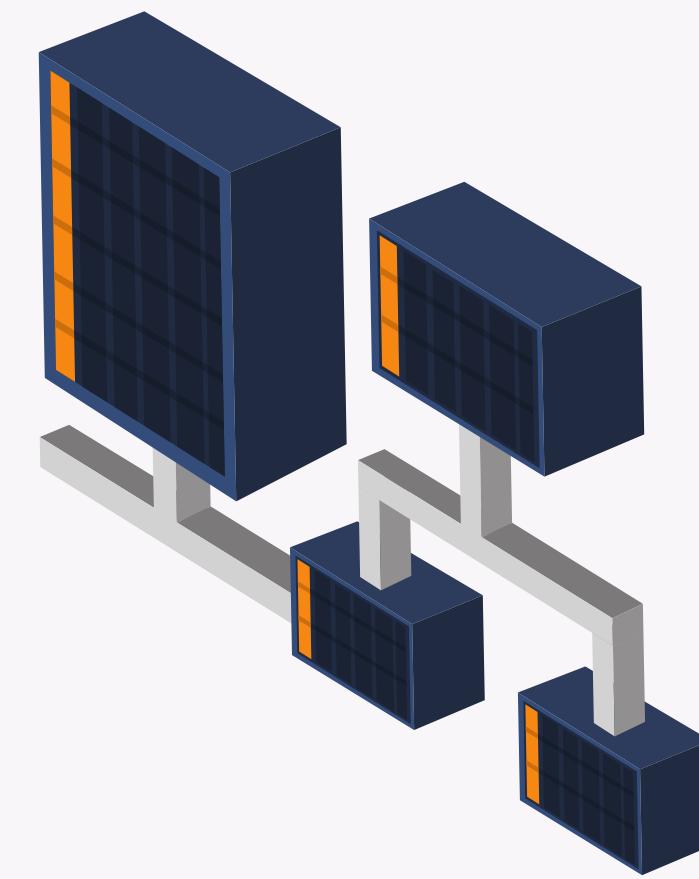
```
auth.dart ×
lib > auth.dart > Auth > accDelete
4
5 class Auth {
6   final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;
7
8   User get currentUser => _firebaseAuth.currentUser;
9
10 Stream<User> get authStateChanges => _firebaseAuth.authStateChanges();
11
12 Future<void> signInWithEmailAndPassword({
13   String email,
14   String password,
15 }) async {
16   await _firebaseAuth.signInWithEmailAndPassword(
17     email: email,
18     password: password,
19   );
20 }
21
22 Future<void> createUserWithEmailAndPassword({
23   String email,
24   String password,
25 }) async {
26   await _firebaseAuth.createUserWithEmailAndPassword(
27     email: email, password: password);
28 }
29
30 Future<void> signOut() async {
31   await _firebaseAuth.signOut();
32 }
33
34 Future<void> accDelete() async {
35   FirebaseFirestore.instance
36     .collection("Users")
37     .doc(FirebaseAuth.instance.currentUser.uid)
38     .delete();
39   await _firebaseAuth.currentUser.delete();
40 }
```

2. In the above code we have created the **login, signout, account delete and register user instances** which will trigger **firebase**, and designed the **frontend layout** for the same

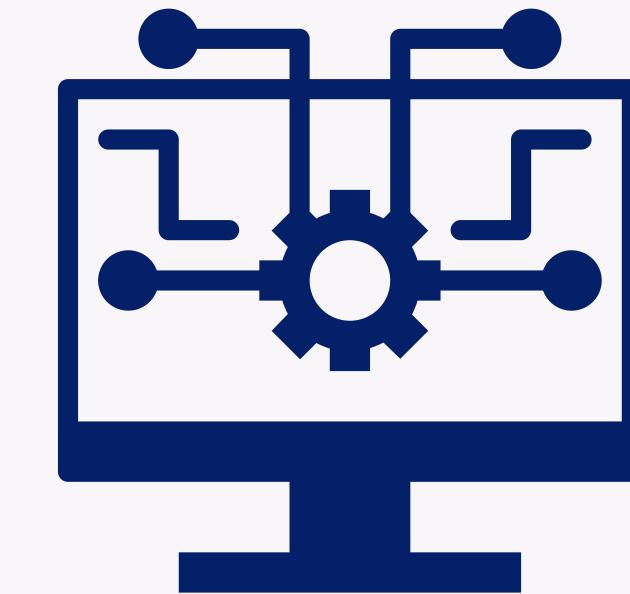
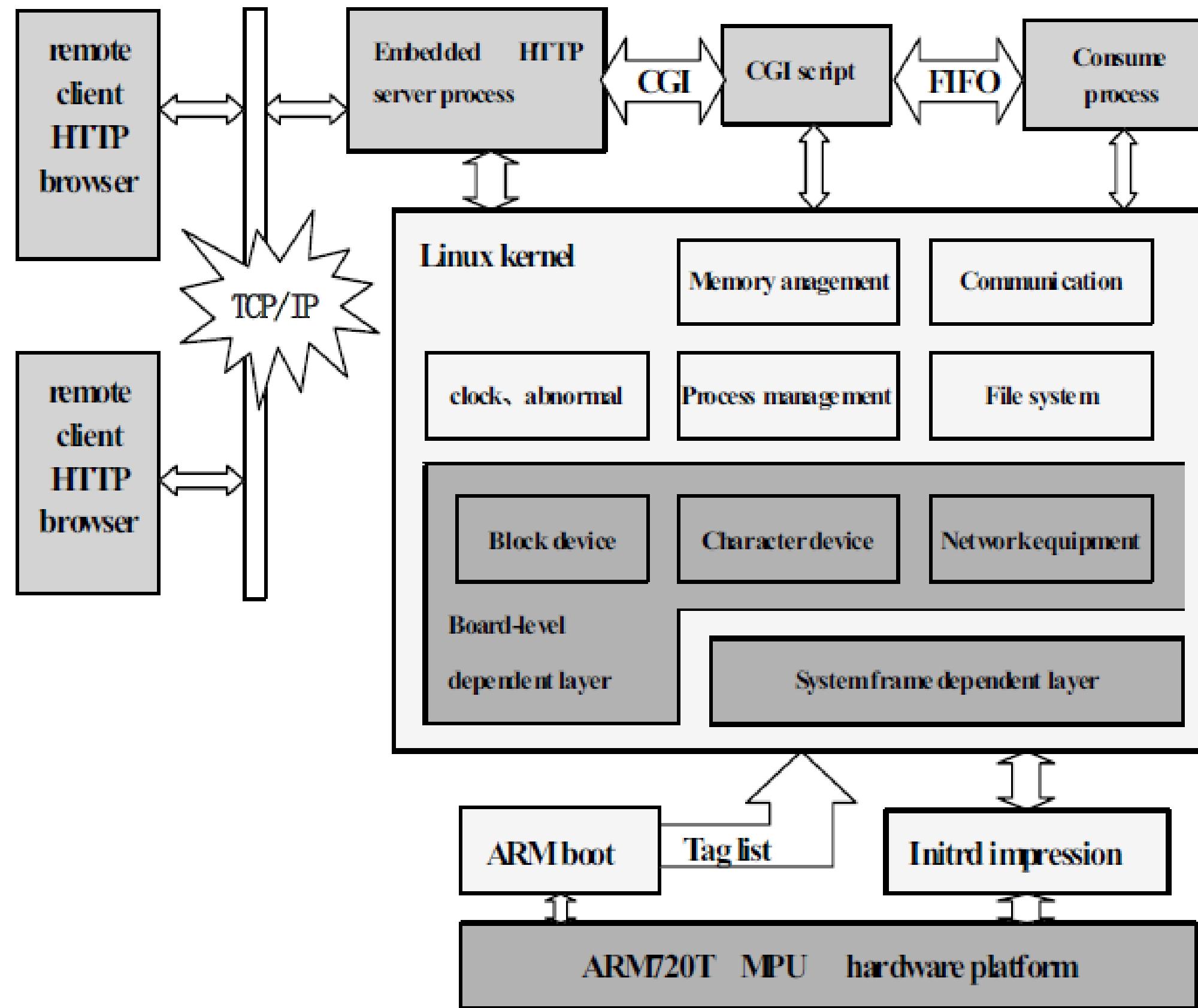
```
terminal_page.dart X

lib > pages > dashboard > screens > product > terminal_page.dart > _BodyState > getOutput

18
19 class _BodyState extends State<Body> {
20   String terminalopt = 'IN_Terminal';
21   final _formKey = GlobalKey<FormState>();
22   final firestoreInstance = FirebaseFirestore.instance;
23   var firebaseUser = FirebaseAuth.instance.currentUser;
24   TextEditingController command = TextEditingController();
25   var input;
26   var url = '';
27   @override
28   void initState() {
29     super.initState();
30     firestoreInstance.collection("Users").doc(firebaseUser.uid).set({
31       "Result": "",
32       "statusCode": "",
33     });
34   }
35
36   getOutput(input) async {
37     if (terminalopt == "IN_Terminal") {
38       url = 'http://43.205.195.2/cgi-bin/cmd.py?x=$input';
39     } else if (terminalopt == "US_Terminal") {
40       url = 'http://3.87.141.161/cgi-bin/cmd.py?x=$input';
41     }
42     var result = await http.get(Uri.parse(url));
43     var output = json.decode(result.body);
44
45     await firestoreInstance.collection("Users").doc(firebaseUser.uid).set({
46       "Result": output['Result'],
47       "statusCode": output['statusCode']
48     }).then((_) => print("Success"));
49   }
}
```



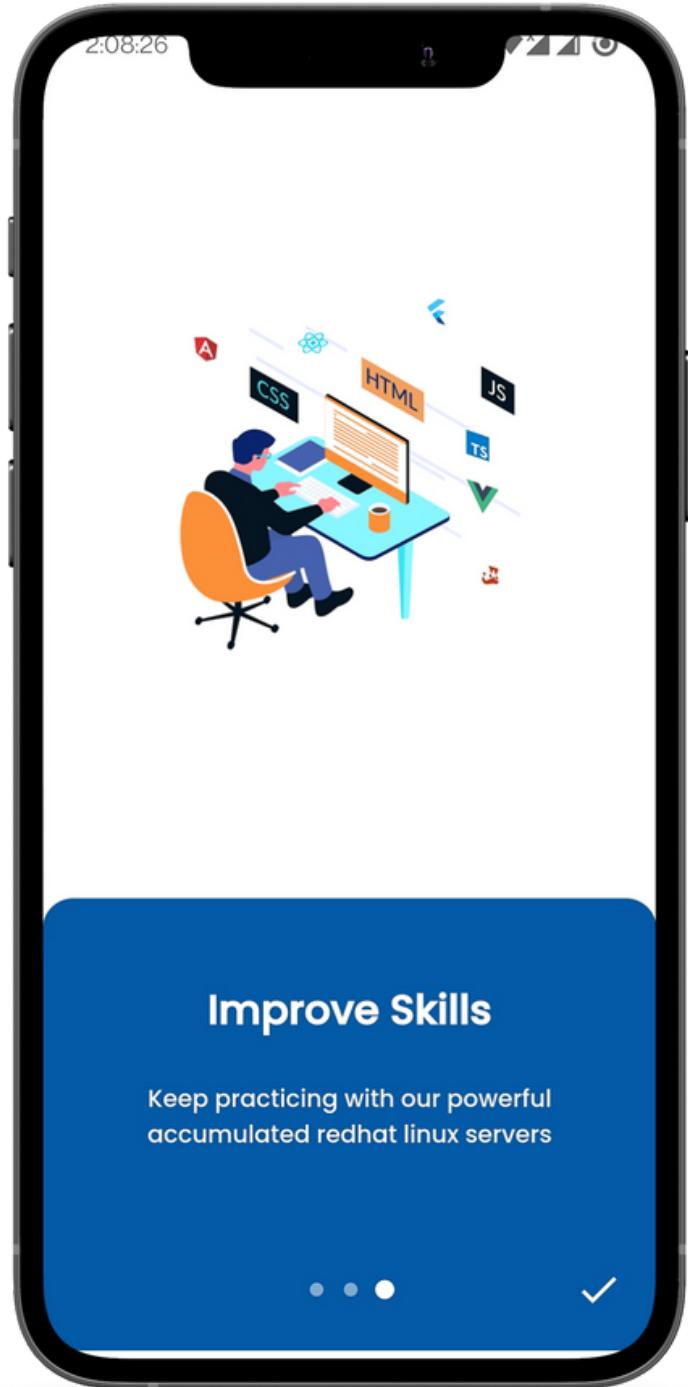
3. In this code we have initiated the command request over **HTTP**, which will be **triggered** to the **IP** we have provided i.e. of our **Linux machines** but with conditions for the **available zones** where we created our machines for user convenience and **load balancing** over the same



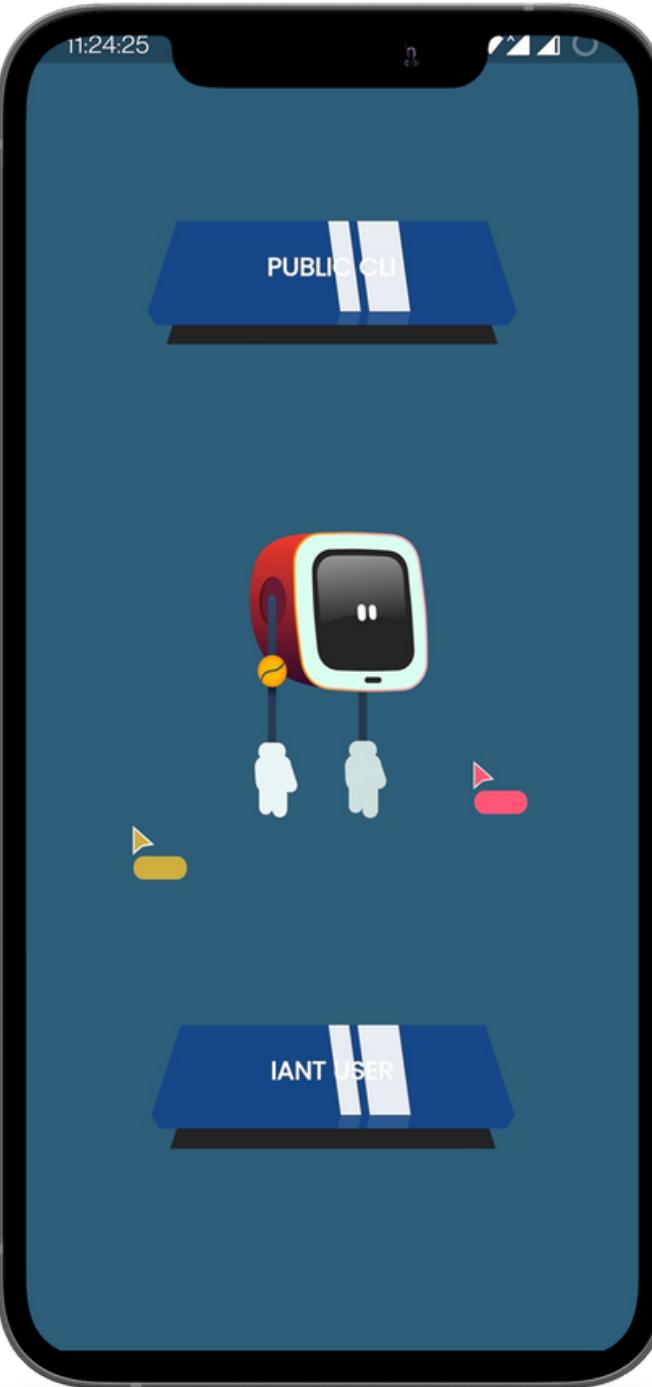
Server architecture and its **internal working** how our **python script** is being converted and processed to the **Linux shell** and upon execution **revert the response** of the same over **TCP/IP protocol** by reliability check on the same



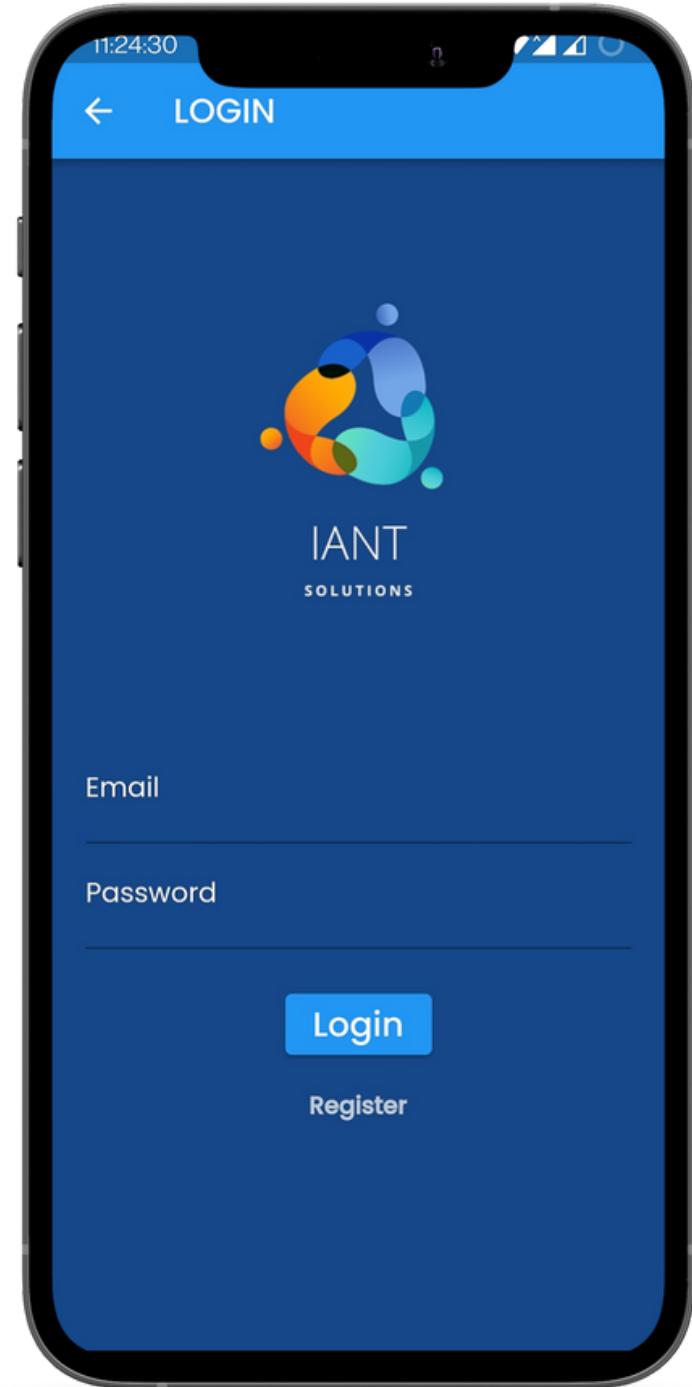
APPLICATION



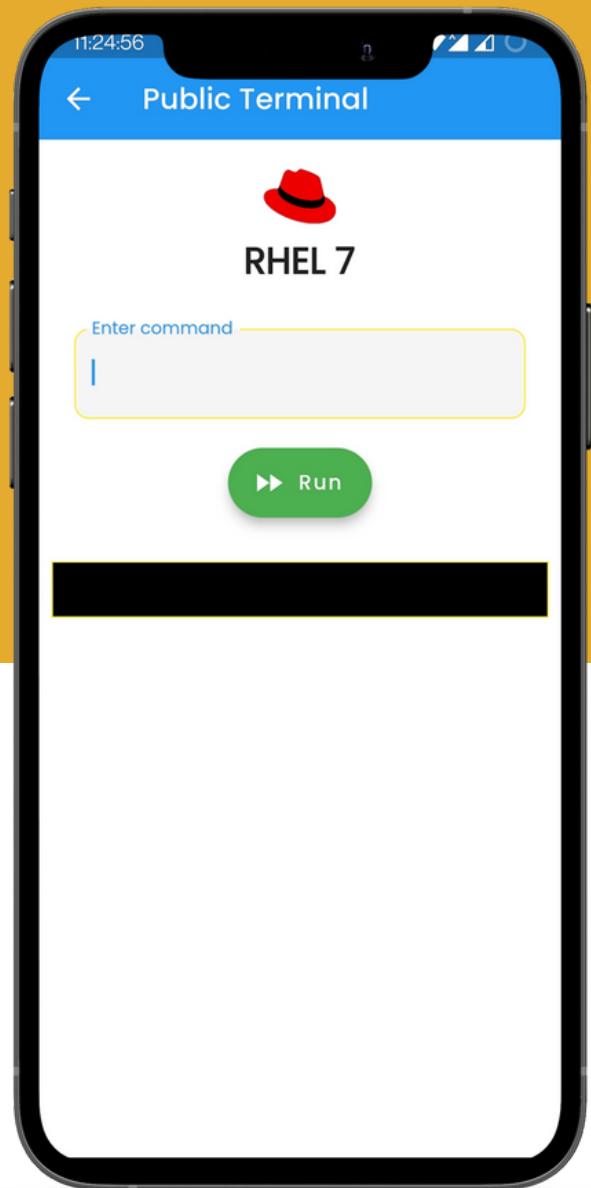
On boarding carousels
for first time launch



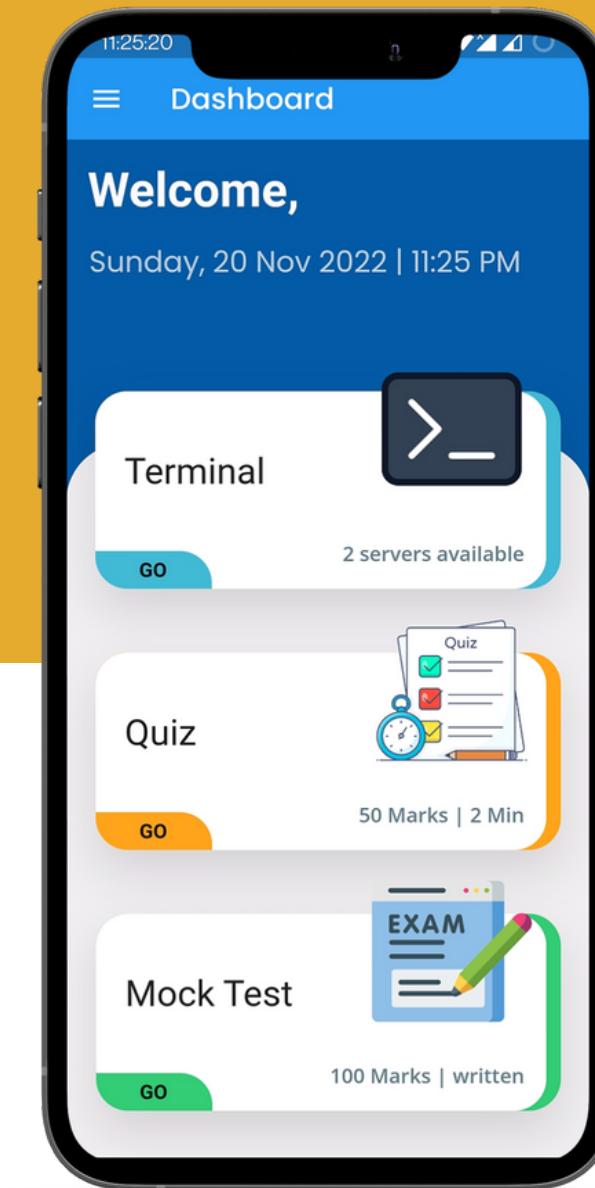
Choice screen selection
done by user



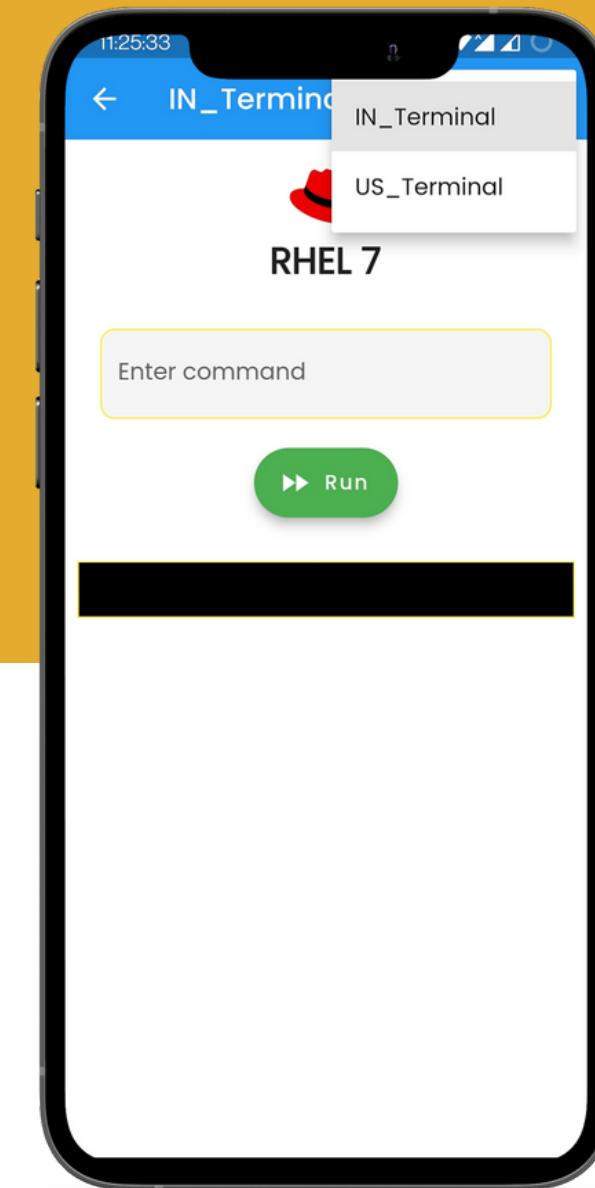
Login screen and register
on the same screen



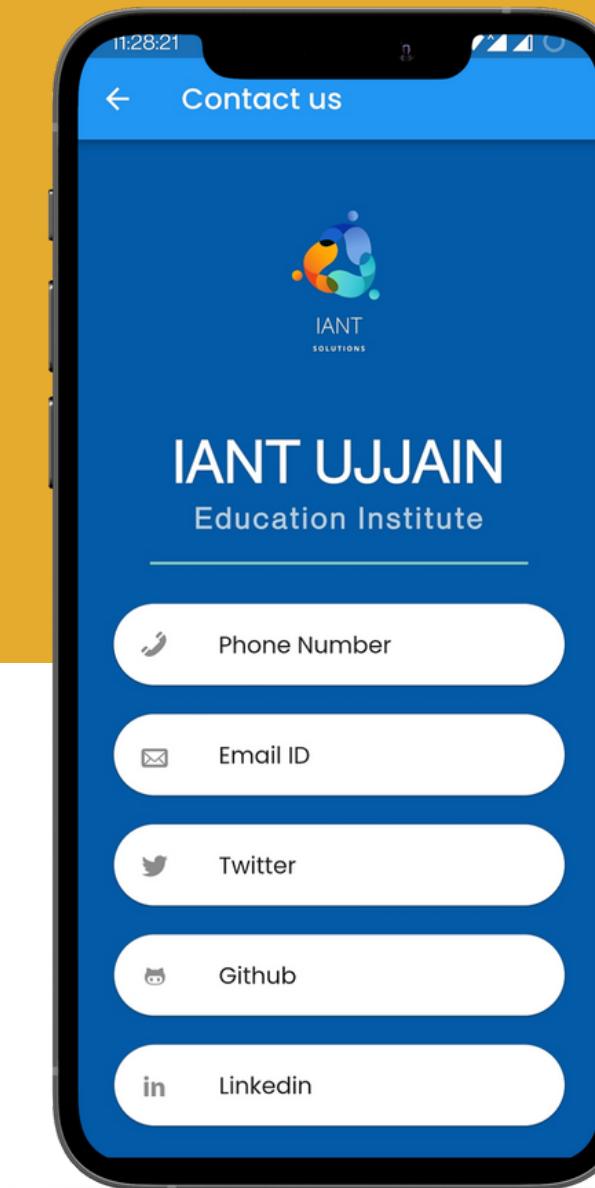
Public terminal used for anytime execution with no restrictions



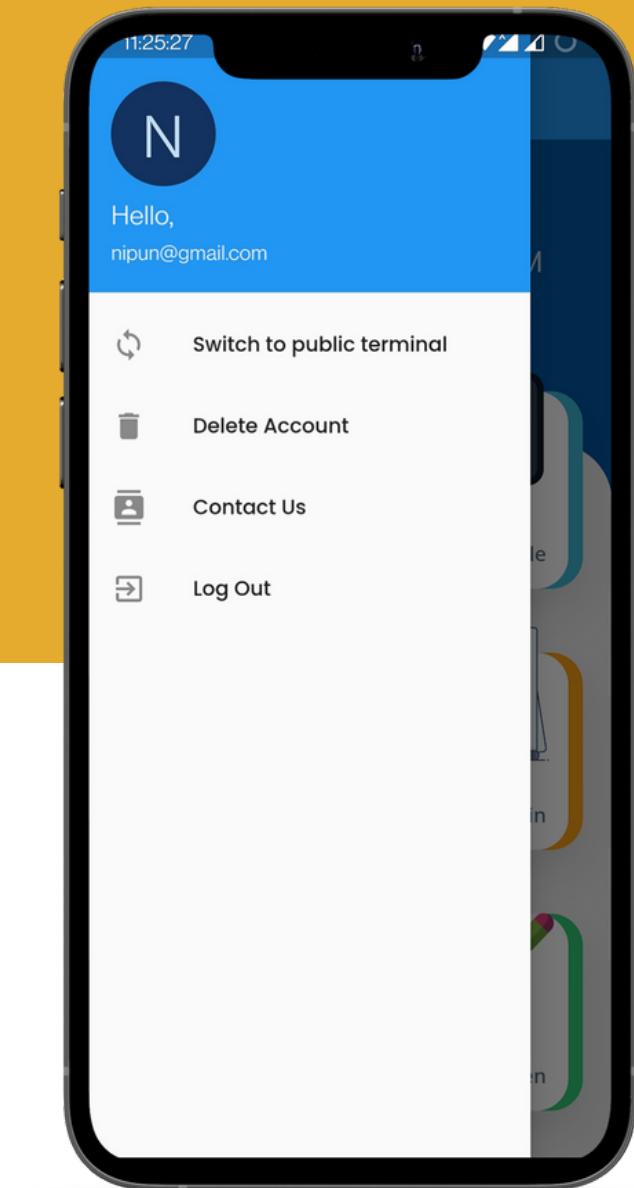
IANT user dashboard on successful login providing quiz and test options



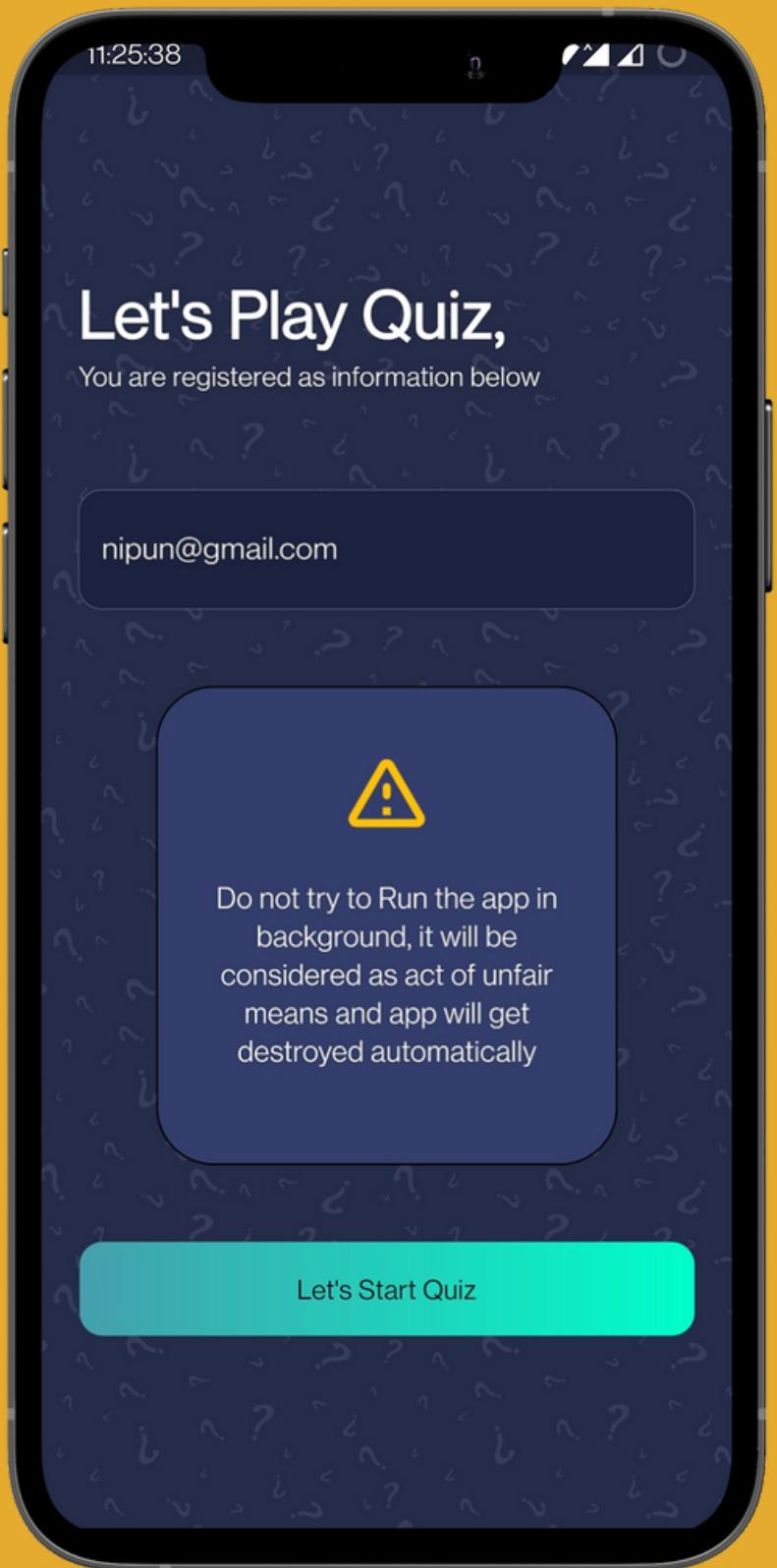
IANT terminal used by users anytime, available in different zones with certain restrictions



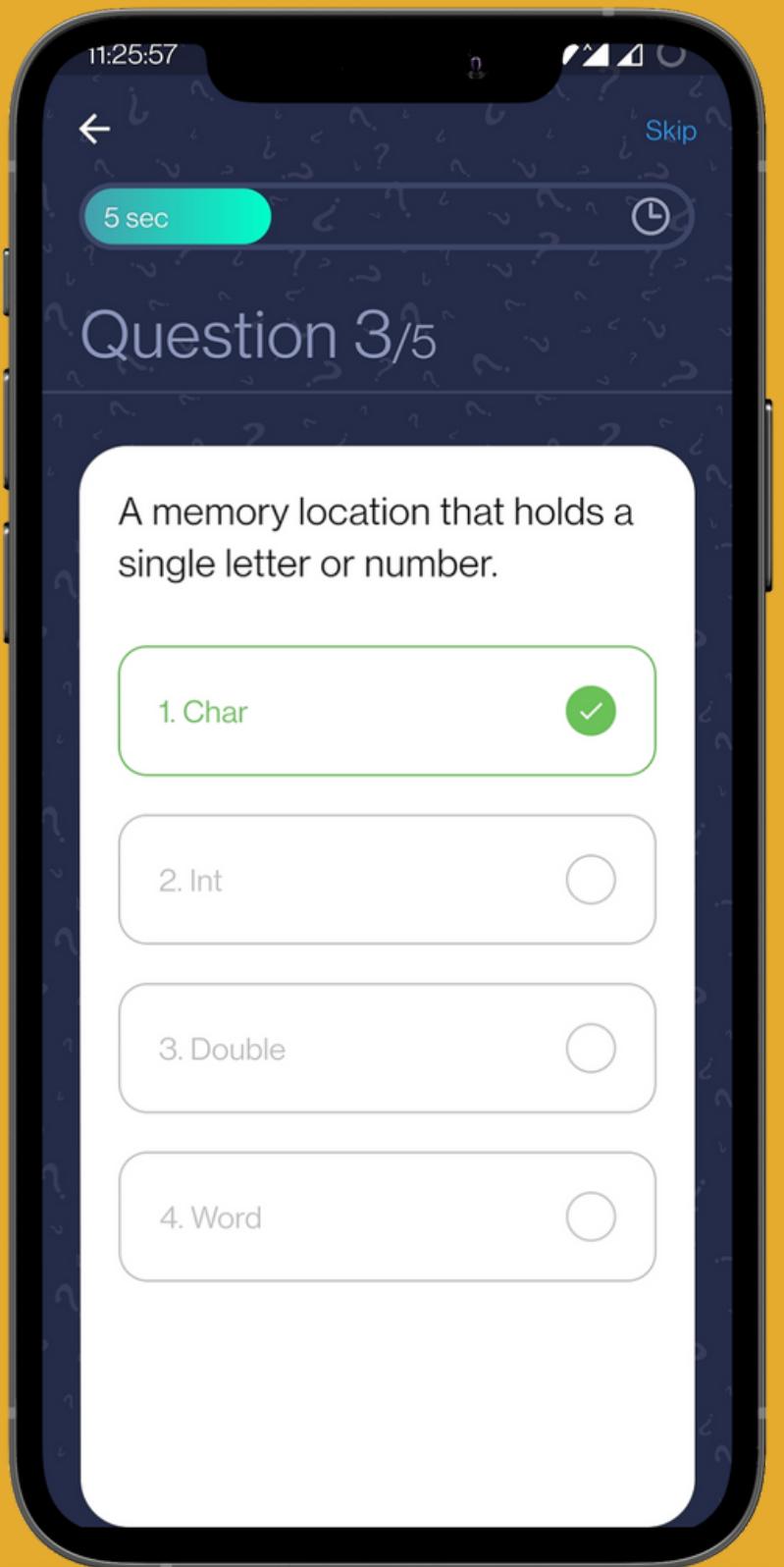
Contact page for anytime accessibility and different links



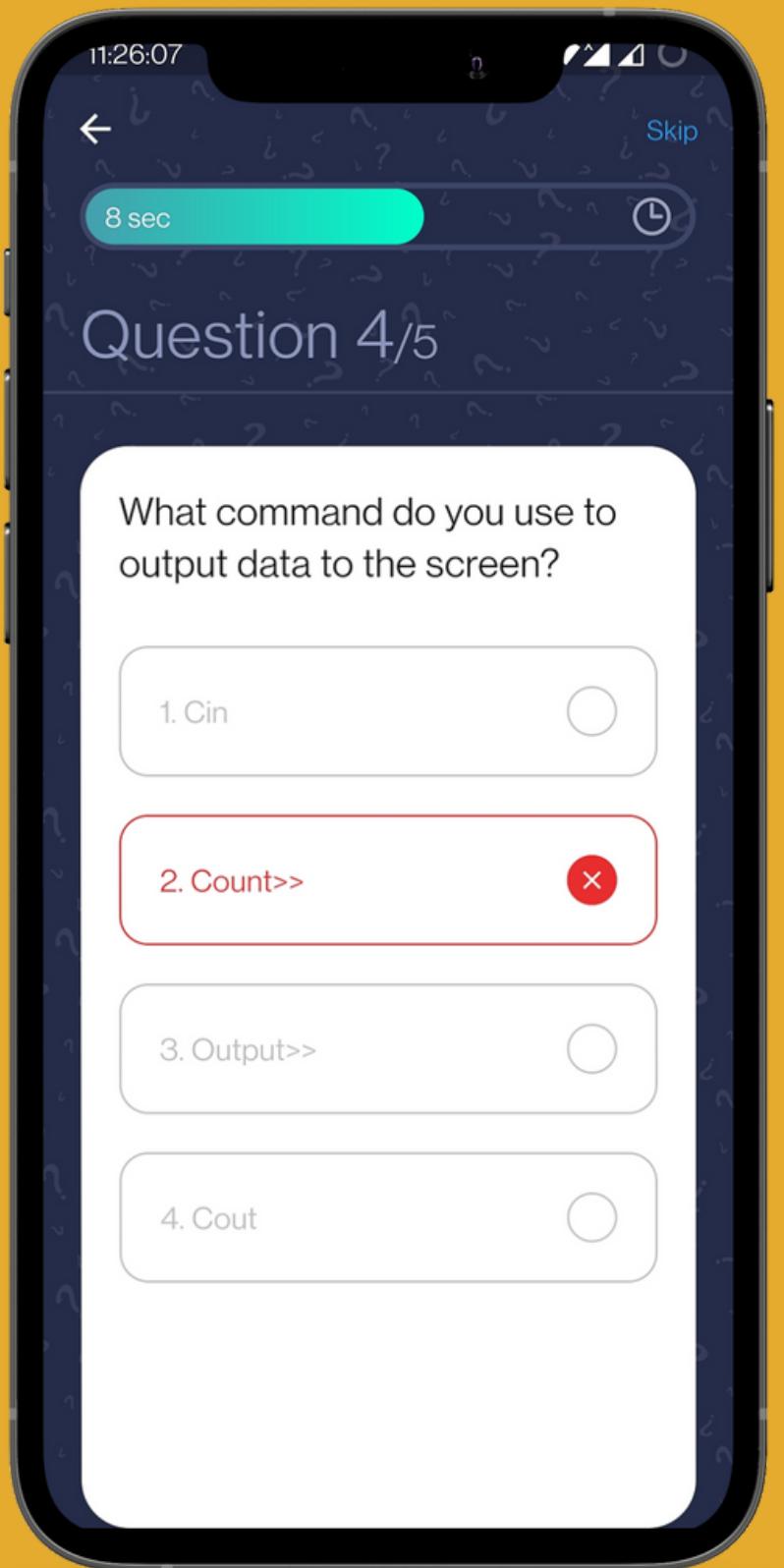
Control page for logout, delete and other features



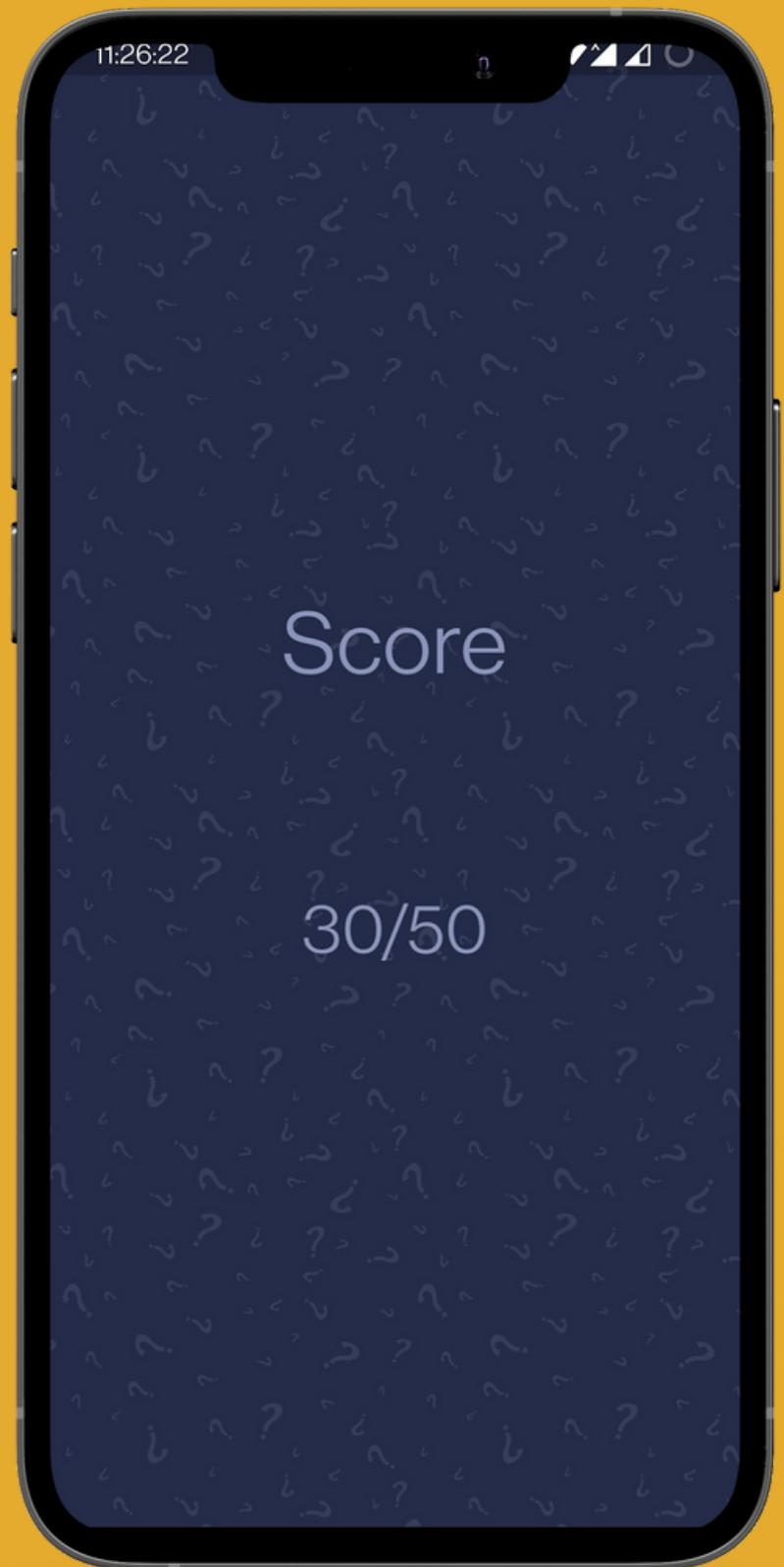
Quiz screen with
warning



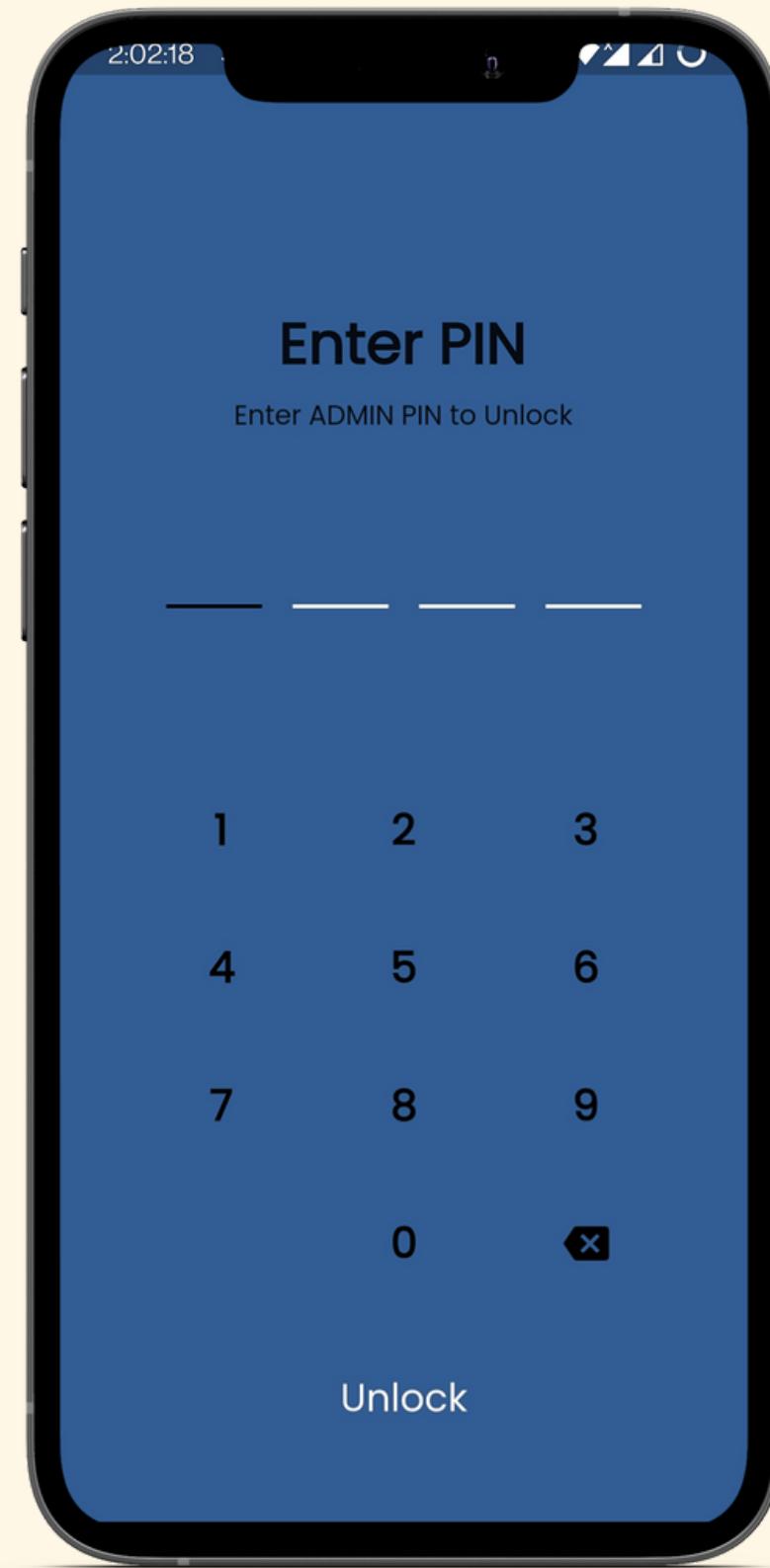
correct answer
selection in time



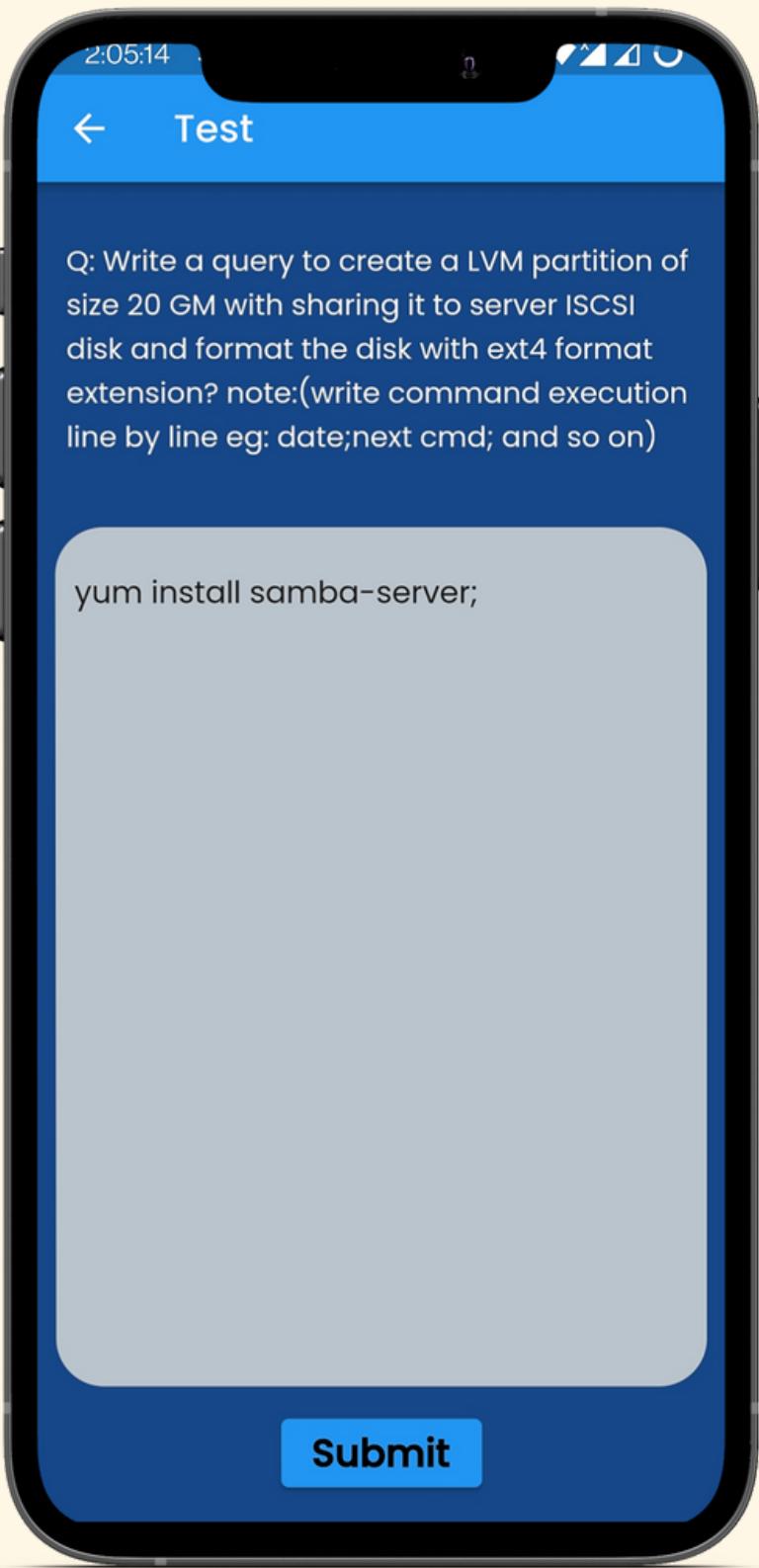
Incorrect answer
with no points



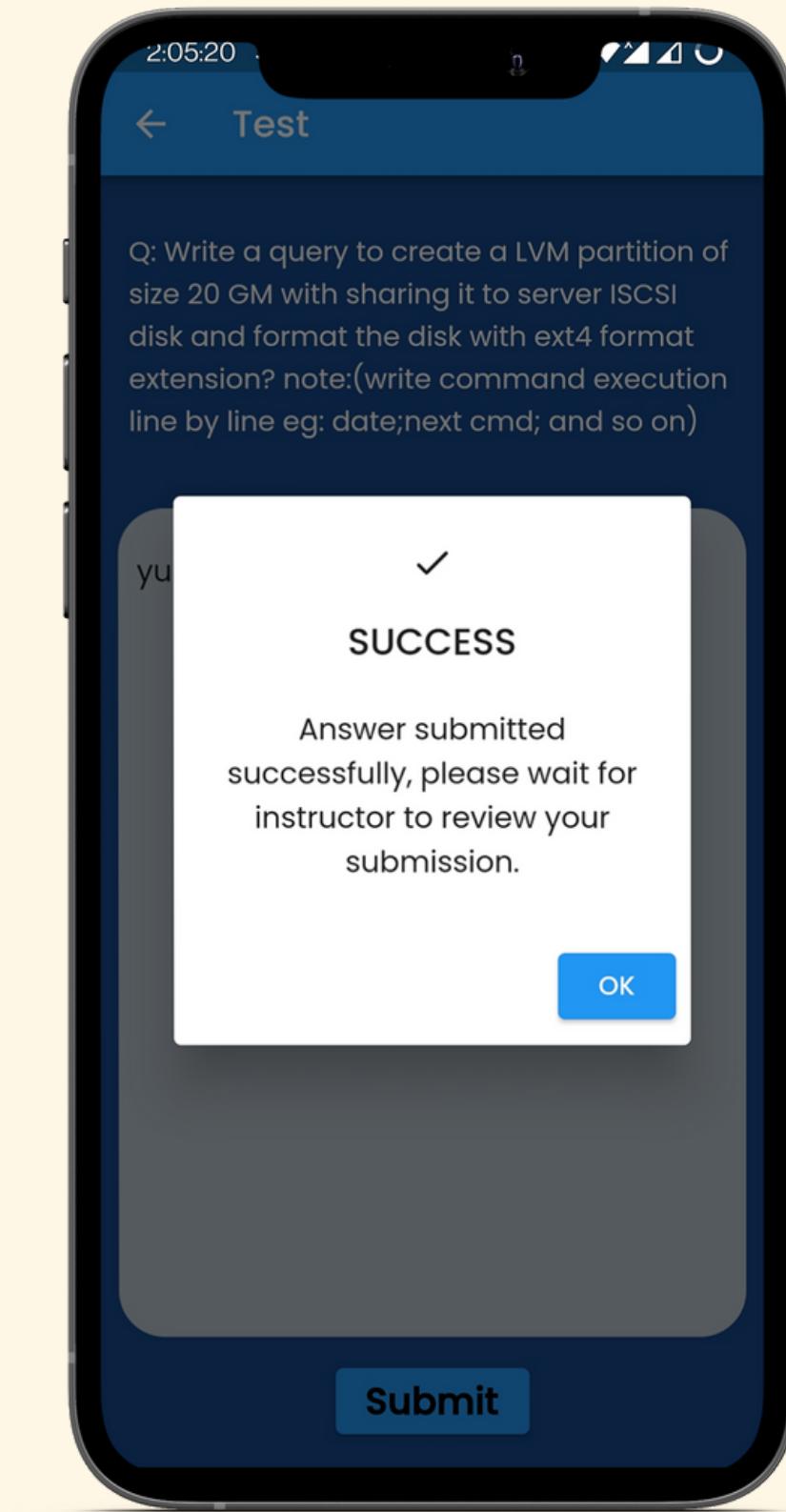
Quiz Score at
successful completion



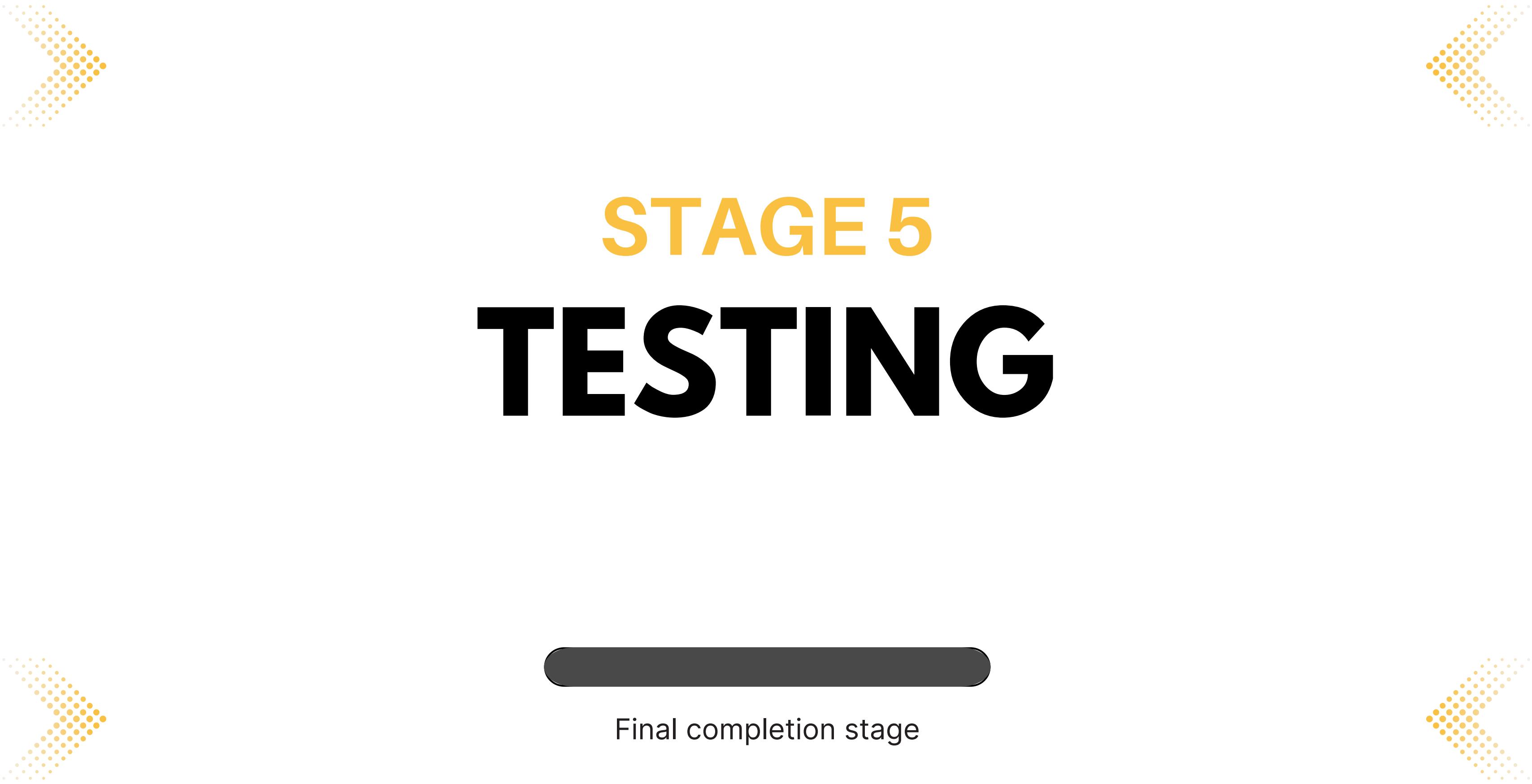
App locked, if tried to run app in background while giving quiz



Test screen, answer to be written for the question and submitted in time



On submission, answer will be sent to IANT database and exit to dashboard



STAGE 5

TESTING



Final completion stage



LOCAL TESTING

Sr No	Test Cases
1	User session handling and data command restoration
2	Quiz app working with background feature app lock
3	User commands and test answers storing in database
4	Login validations working perfectly
5	app responsiveness working perfectly
6	exceptional handling crash handling

Case 1: User session handling and data command restoration

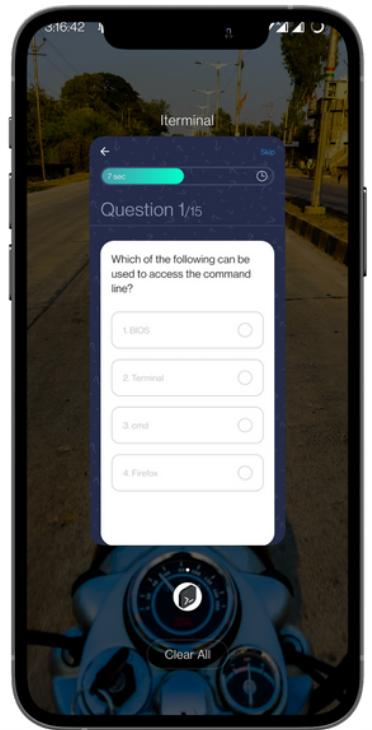
whenever we try to log in once and close the app without logging in the data remains saved automatically in the firebase snapshot and upon relaunch of the app, the data of the lost login user on that dedicated device gets restored and synced in the same state.

STATUS

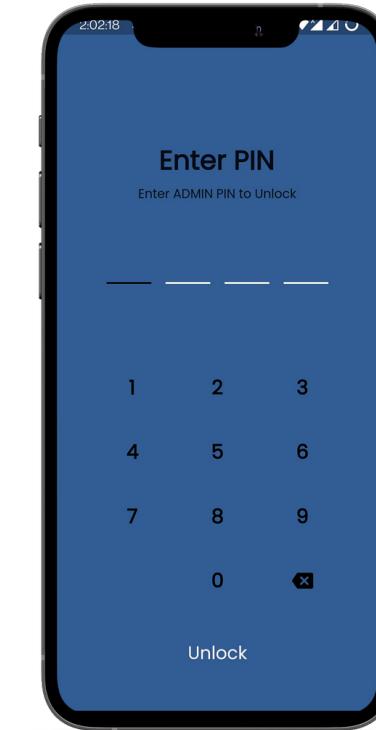
Passed



Case 2: Quiz app working with background feature app lock



Yes the apps getting crashed and getting locked unless the correct password is granted only when you launch the quiz and try to go to background in the mobile phone



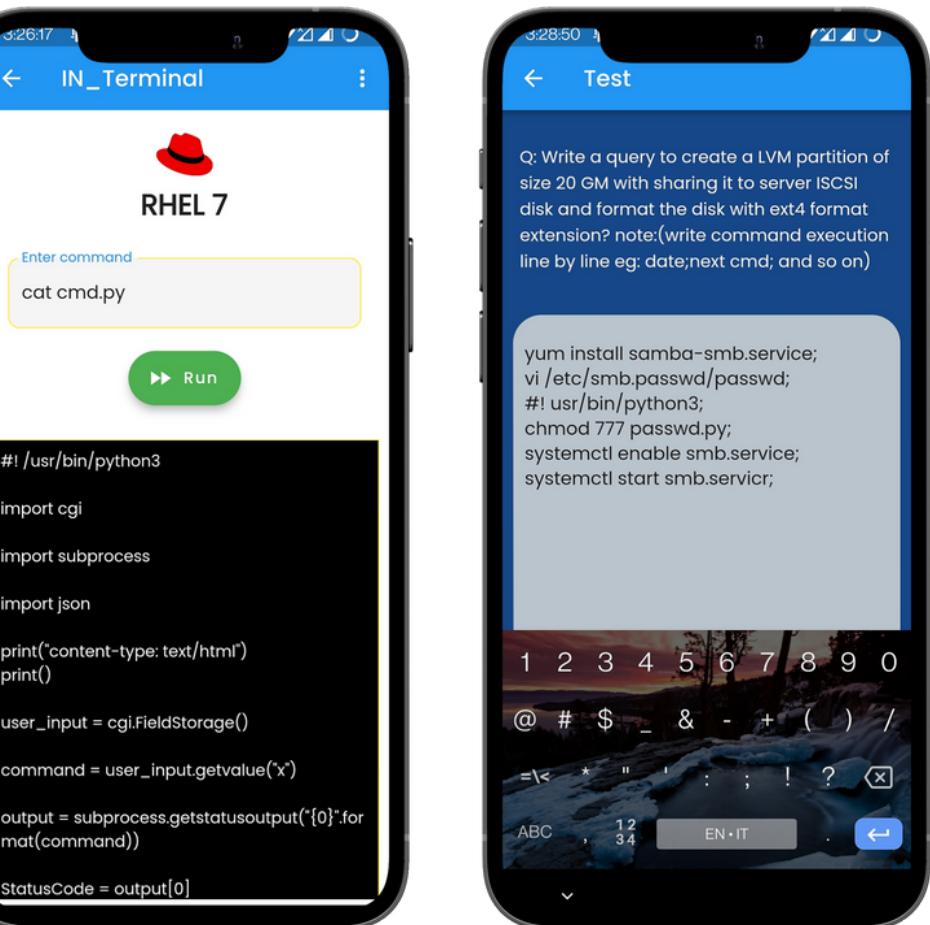
Passed



Case 3: User commands and test answers storing in database

Yes the commands and user answers are getting stored on a database and blank submissions are not accepted, the exception is working accurately, since the database is not real-time, so it takes to reflect the data at most of 5 seconds of delay at the backend

Passed

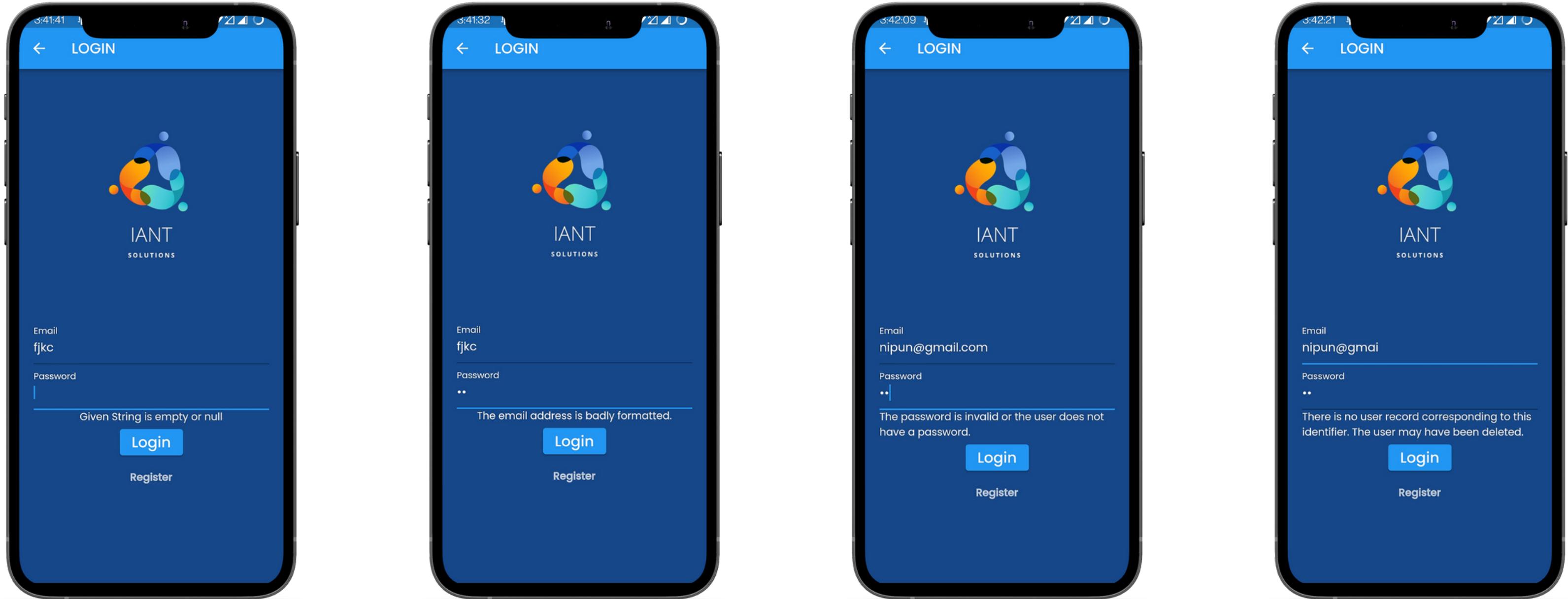


```
Result: "#!/usr/bin/python3 import cgi import subprocess import json print('content-type: text/html') print() user_input = cgi.FieldStorage() command = user_input.getvalue('x') output = subprocess.getstatusoutput('{0}'.format(command)) StatusCode = output[0] result = output[1] db = { 'Result': result, 'StatusCode': StatusCode } finalResult = json.dumps(db) print(finalResult)" StatusCode: 0
```

```
Result: "#!/usr/bin/python3 import cgi import subprocess import json print('content-type: text/html') print() user_input = cgi.FieldStorage() command = user_input.getvalue('x') output = subprocess.getstatusoutput('{0}'.format(command)) StatusCode = output[0] result = output[1] db = { 'Result': result, 'StatusCode': StatusCode } finalResult = json.dumps(db) print(finalResult)" StatusCode: 0
```

the reflection of data is seen in the database of firebase service as well as seen in the mobile application

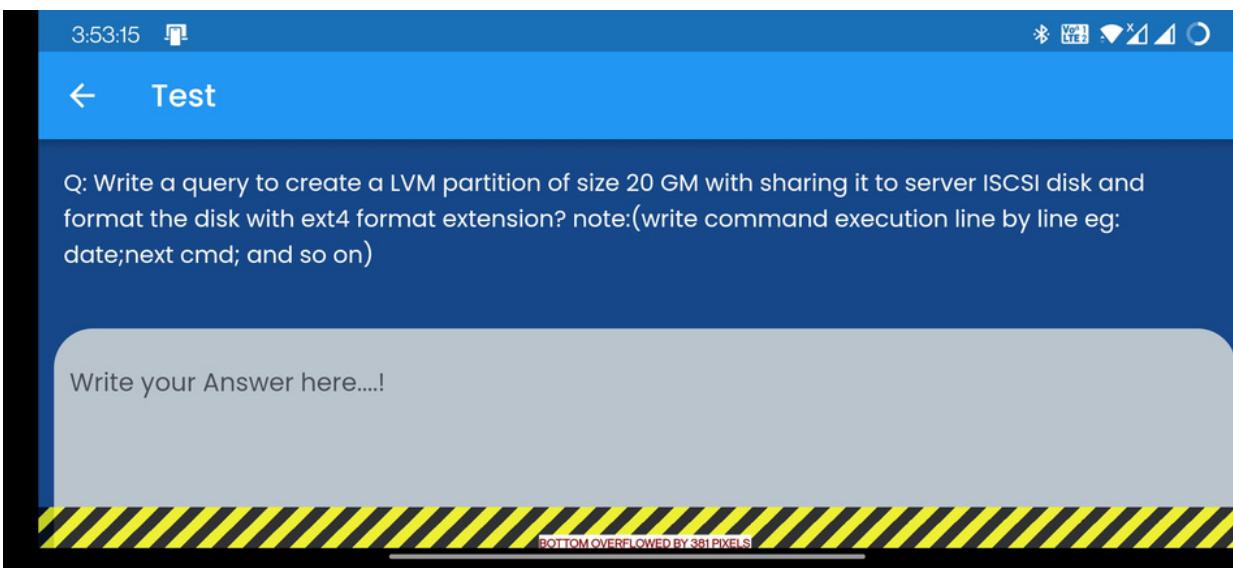
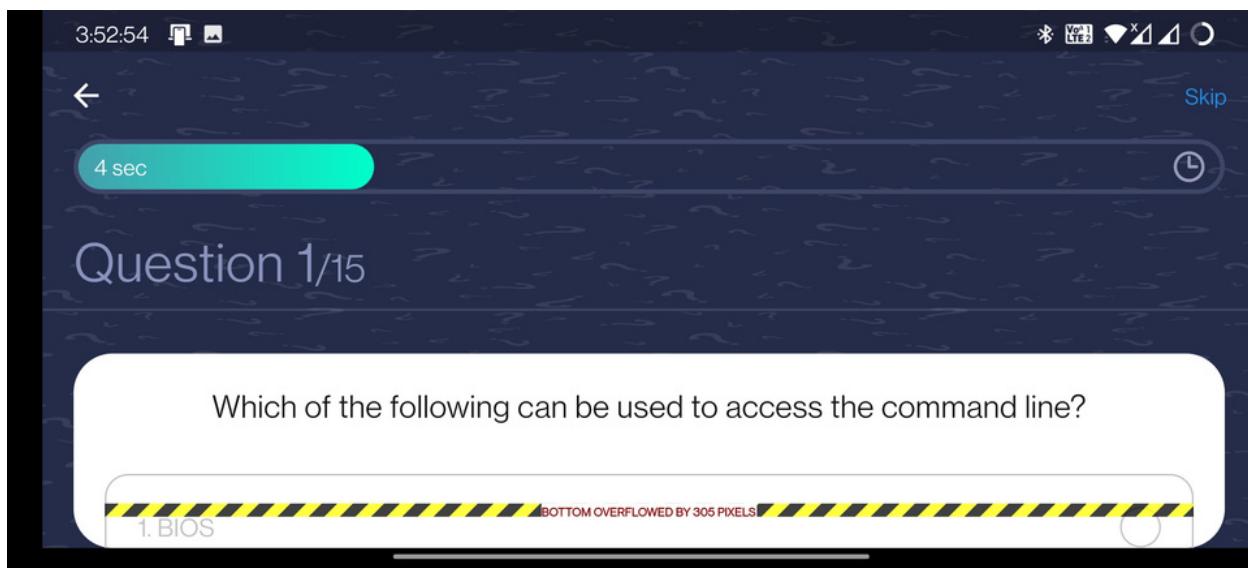
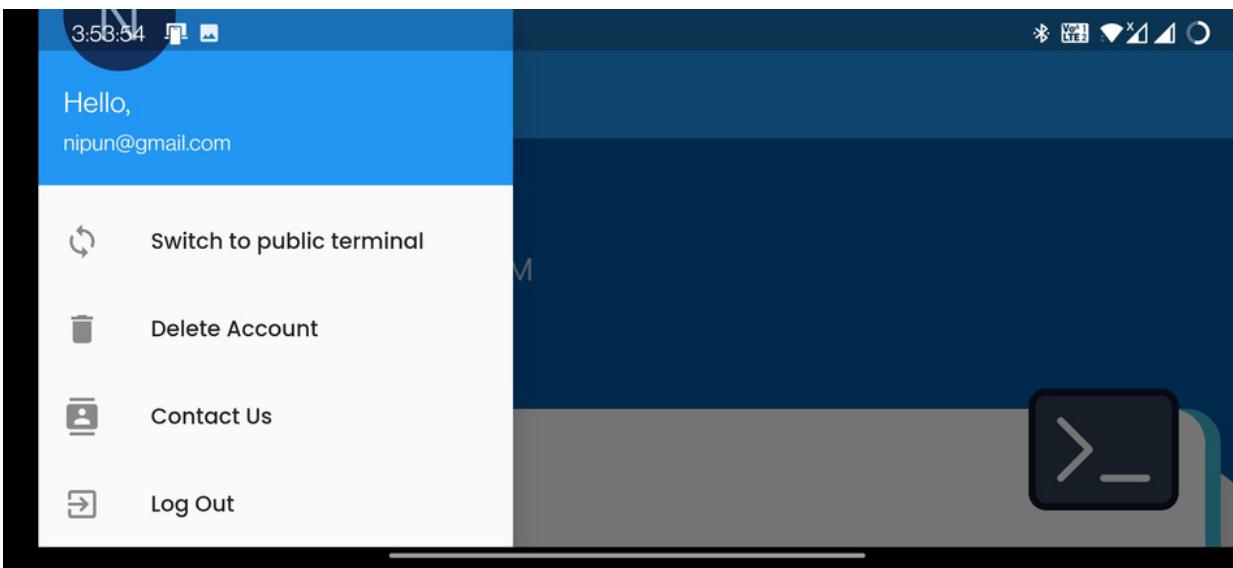
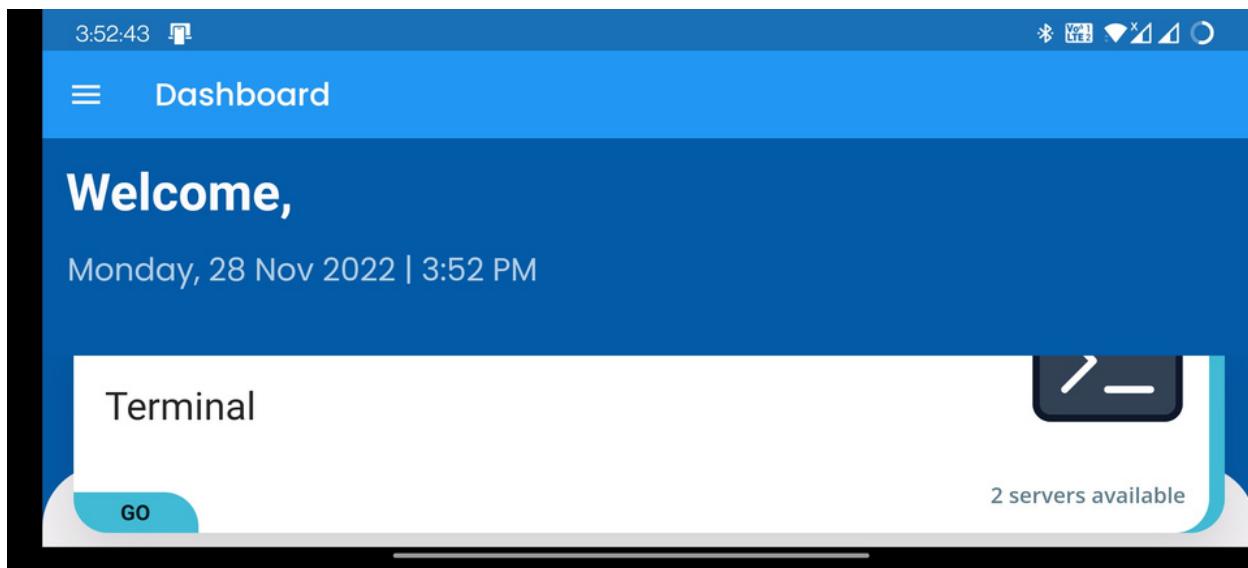
Case 4: Login validations working perfectly



All the validations are working perfectly, from providing the empty strings to providing wrong passwords as well as non-existing user details in the credentials is handled by inbuilt firebase validation feature

Passed 

Case 5: App responsiveness working perfectly



There are certain orientation glitches in different sections of dashboards such as pixel overflow found in the test and quiz screen while responsiveness is handled in dashboards and drawer areas.

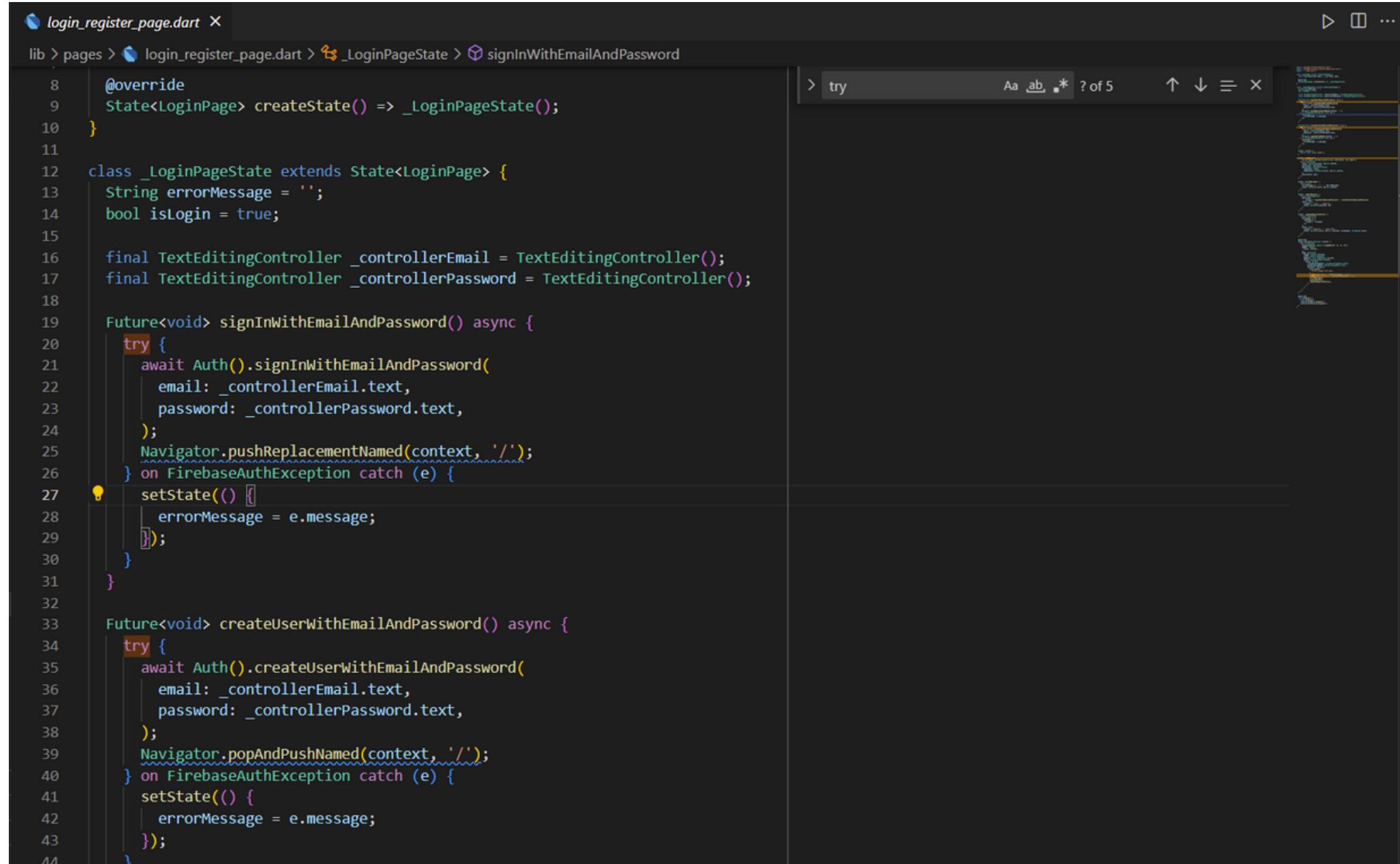
Failed 

Resoultion

Since in order to correct the layout for different styles we tried flutter's media query statement which takes only as much space required as the size of the device displaying the output of the application, in this, we way it can be achieved

Case 6: exceptional handling crash handling

The exceptional handling is taken care of and is working perfectly we have defined each and every firebase authentication call and database trigger into a **try-catch** block in order to avoid the app crashing which is seen already in the quiz background lock state and authentication state of the login page.



```
lib > pages > login_register_page.dart > _LoginPageState > signInWithEmailAndPassword
8   @override
9     State<LoginPage> createState() => _LoginPageState();
10 }
11
12 class _LoginPageState extends State<LoginPage> {
13   String errorMessage = '';
14   bool isLoggedIn = true;
15
16   final TextEditingController _controllerEmail = TextEditingController();
17   final TextEditingController _controllerPassword = TextEditingController();
18
19   Future<void> signInWithEmailAndPassword() async {
20     try {
21       await Auth().signInWithEmailAndPassword(
22         email: _controllerEmail.text,
23         password: _controllerPassword.text,
24       );
25       Navigator.pushReplacementNamed(context, '/');
26     } on FirebaseAuthException catch (e) {
27       setState(() {
28         errorMessage = e.message;
29       });
30     }
31   }
32
33   Future<void> createUserWithEmailAndPassword() async {
34     try {
35       await Auth().createUserWithEmailAndPassword(
36         email: _controllerEmail.text,
37         password: _controllerPassword.text,
38       );
39       Navigator.popAndPushNamed(context, '/');
40     } on FirebaseAuthException catch (e) {
41       setState(() {
42         errorMessage = e.message;
43       });
44     }
45   }
46 }
```

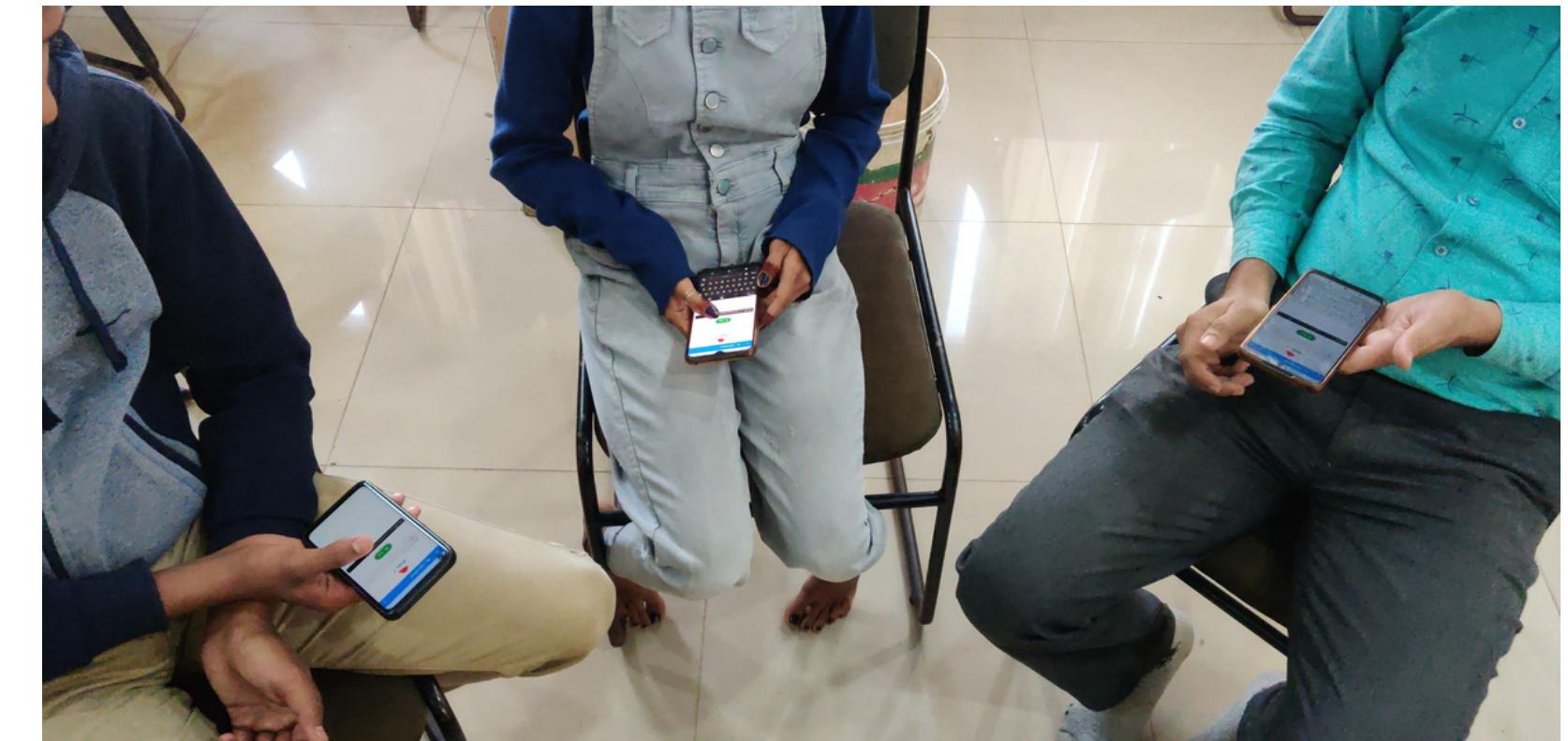
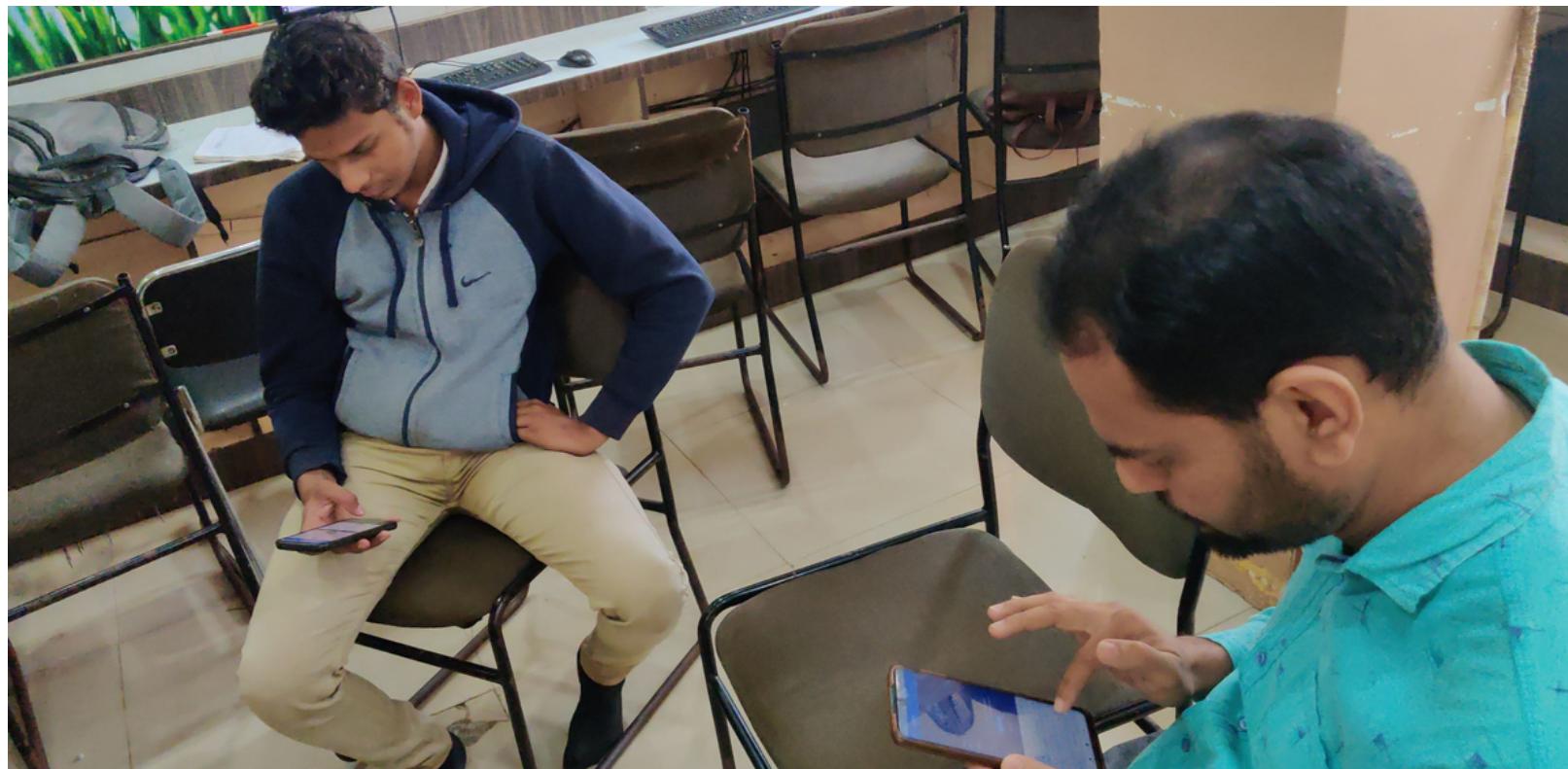
Passed





CLIENT TESTING

Case: Testing with multiple clients on the same terminal with multiple users on it



We have tested with the client and performed many operations from basic to complex from different zones as well as with the same zone so the output is coming out to perfection in many cases complex queries need resolution if possible but as AWS constraint rectification require charges to configure.

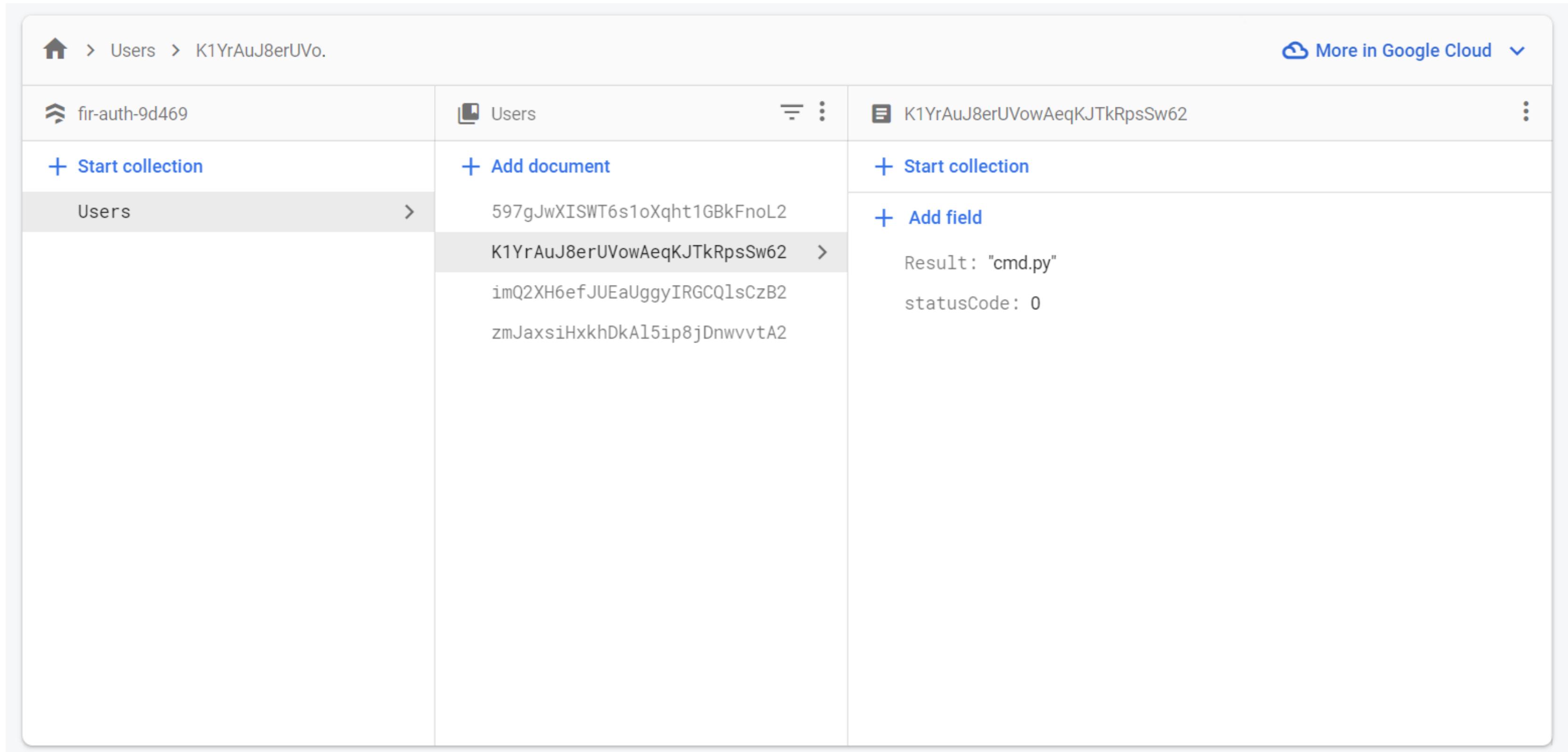
STATUS

Moderate



Case: Checking the load balancing/ machine performance

UNCLEAR 



The screenshot shows the Google Firestore console interface. The path in the top left is `> Users > K1YrAuJ8erUVo.`. The top right has a "More in Google Cloud" button. The left sidebar shows a project named "fir-auth-9d469" and a "Users" collection. The main area displays a document with the ID "K1YrAuJ8erUVo". The document contains two fields:

- `Result: "cmd.py"`
- `statusCode: 0`

The screenshot shows two document snapshots in the Google Cloud Firestore console. The top document, with ID `imQ2XH6efJUEaUuggyIRGCQlsCzB2`, has fields:

- `Result`: "Thu Dec 1 07:10:38 UTC 2022"
- `Zone`: "INDIA (Mumbai ap-south-1)"
- `statusCode`: 0

The bottom document, with ID `K1YrAuJ8erUVowAeqKJTkRpsSw62`, has fields:

- `Result`: "Thu Dec 1 07:12:00 UTC 2022"
- `Zone`: "USA (N.Virginia us-east-1)"
- `statusCode`: 0

For the load balancing part, we doesn't have a sufficient no. of users to test upon but for the machine performance we checked with a few users, and doesn't seem to have any issues in speed or performance

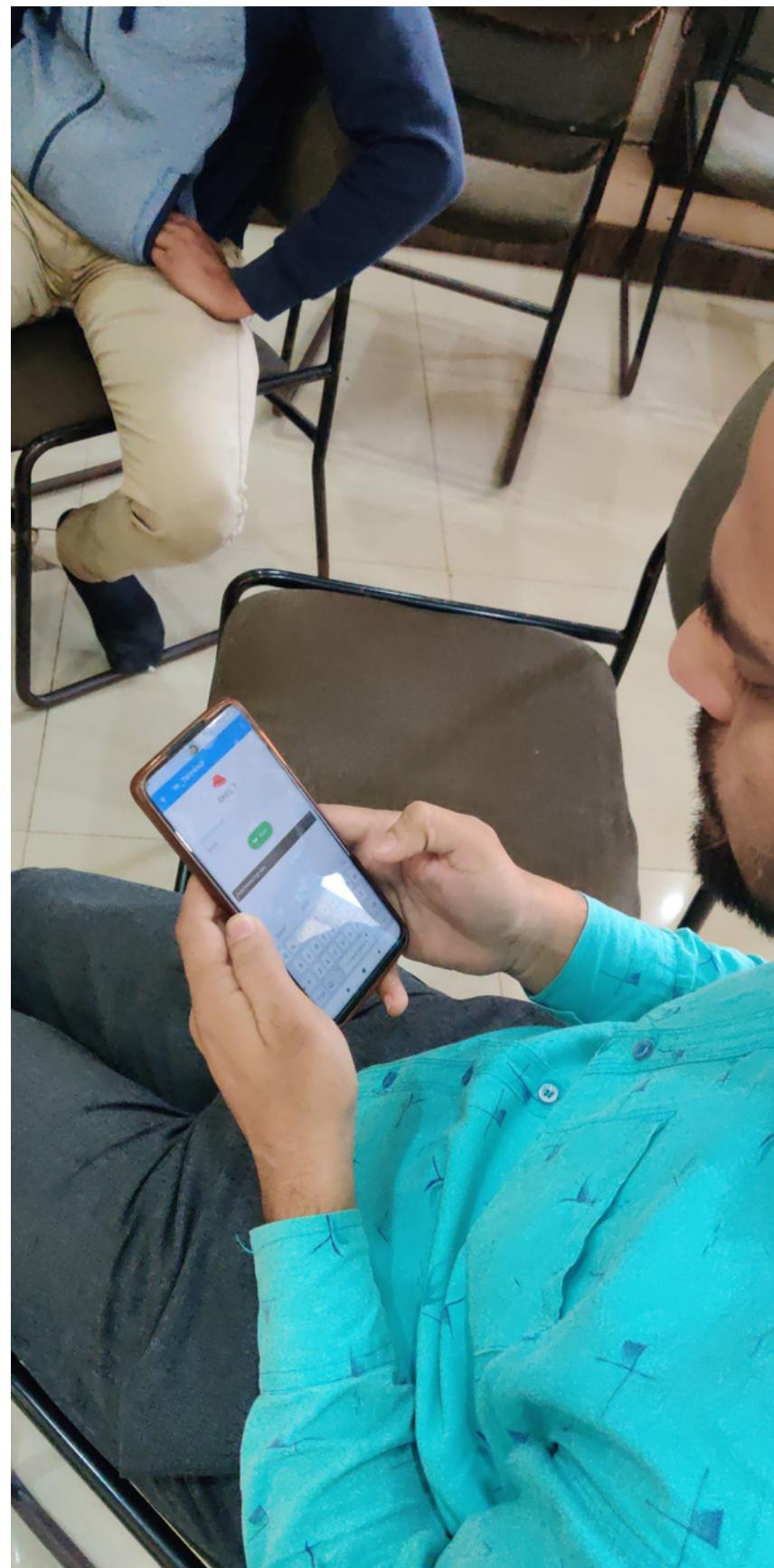
The screenshot shows two document snapshots in the Google Cloud Firestore console. The top document, with ID `imQ2XH6efJUEaUuggyIRGCQlsCzB2`, has fields:

- `Result`: "Thu Dec 1 07:10:38 UTC 2022"
- `Zone`: "INDIA (Mumbai ap-south-1)"
- `statusCode`: 0

The bottom document, with ID `K1YrAuJ8erUVowAeqKJTkRpsSw62`, has fields:

- `Result`: "Thu Dec 1 07:12:00 UTC 2022"
- `Zone`: "USA (N.Virginia us-east-1)"
- `statusCode`: 0

As per the load criteria we have defined zones in the database in order to identify the user working on which zone terminal



IN_Terminal

RHEL 7

Enter command

```
yum install samba -y
```

Run

Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered with an entitlement server. You can use subscription-manager to register.
Last metadata expiration check: 4:07:53 ago on Wed Nov 30 09:39:46 2022.
Dependencies resolved.
=====
=====
=====
Package Arch Version
Repository Size
=====
=====
=====

IN_Terminal

```
15/22 Verifying : libldb-2.4.1-1.el8.x86_64  
20/22 Verifying : libtdb-1.4.6-1.el8.x86_64  
21/22 Verifying : libtdb-1.4.4-1.el8.x86_64  
22/22 Installed products updated.  
  
Upgraded:  
libldb-2.5.2-2.el8.x86_64  
libtdb-1.4.6-1.el8.x86_64  
Installed:  
avahi-libs-0.7-20.el8.x86_64      cups-  
libs-1:2.2.6-50.el8.x86_64  
libicu-60.3-2.el8_1.x86_64  
libsmbclient-4.16.4-2.el8.x86_64  
libwbclient-4.16.4-2.el8.x86_64      python3-  
dns-1.15.0-11.el8.noarch  
python3-lldb-2.5.2-2.el8.x86_64  
python3-samba-4.16.4-2.el8.x86_64  
python3-talloc-2.3.3-1.el8.x86_64  
python3-tdb-1.4.6-1.el8.x86_64  
python3-tevent-0.11.0-0.el8.x86_64  
samba-4.16.4-2.el8.x86_64  
samba-client-libs-4.16.4-2.el8.x86_64  
samba-common-4.16.4-2.el8.noarch  
samba-common-libs-4.16.4-2.el8.x86_64  
samba-common-tools-4.16.4-2.el8.x86_64  
samba-libs-4.16.4-2.el8.x86_64      tdb-  
tools-1.4.6-1.el8.x86_64  
  
Complete!
```

RESULT ANALYSIS

Hence, upon testing out the test cases criteria which we presented in the table came upon the result score of 5/6 which summarize the whole project as below,

user session handling is passed in all cases as the data is saved in the firebase snapshot which gets recovered upon app relaunch, also the quiz feature is successfully working passing all three parameters that are getting locked while running in the background, only the admin password can unlock that app upon successful login boolean is set again back to initial criteria's

all the last user's data is being monitored and saved in the database as per the client-given requirement and ACL permissions are working perfectly for the user to protect the server security, all the exceptions are also handled for login or bad gateway requests passed such as null values in a command terminal, etc.

only the responsiveness of our app in different screen sizes is not passed which was resolved to some extent using a media query, but exceptions were there for foldable type devices.

APPLICATION LIMITATIONS

0
1

CGI CONSTRAINT

We cannot perform high end command configurations as CGI doesn't support it

0
2

FIREBASE MONITOR

Firebase provides a monitor for limited no. of logins, beyond that it is a chargeable facility

0
3

AWS LIMITATION

cannot run multiple EC2 instance as free tier offers only 750 free hours running

0
4

LOAD BALANCING

It gets difficult to identify load zone when multiple user work on same EC2 zone

0
5

SCALABILITY

Scaling this application will cost server maintenance charges



DEPLOYMENT

The screenshot shows a GitHub repository page for 'nipunpatel27 / Linux---Flutter-App'. The repository is public. The 'Code' tab is selected. The main content area displays a single file, 'README.md', which contains the text 'Linux---Flutter-App'. The repository has 1 branch and 0 tags. There are two commits from 'nipunpatel27' with the message 'Create README.md'. The latest commit was made 23 minutes ago.

[Click to download app now](#)

The application is deployed on the GitHub platform from my repository it can be downloaded.

As the play store requires fees for deploying applications other verification constraints diverted us to deploy the application free on GitHub, providing easy accessibility.



CONCLUSION AND FUTURE SCOPE

Hence we have successfully completed all stages from designing to implementing and testing the application, during the client-side testing phase we are not able to complete a few of the test cases as it requires a bulk of users to test the application at the same time i.e. load balancing apart from it, features that were discussed during ideation are implemented with the completion rate of 95%.

Future Scope:

- 1.) Auto suggestion of commands can also be done using ML.**
- 2.) Pre-defined course access can be given for some fees.**
- 3.) Mock test can be expanded for multiple no. of questions with a time limit.**
- 4.) Google sign-in can be added for the Iant user convenience.**