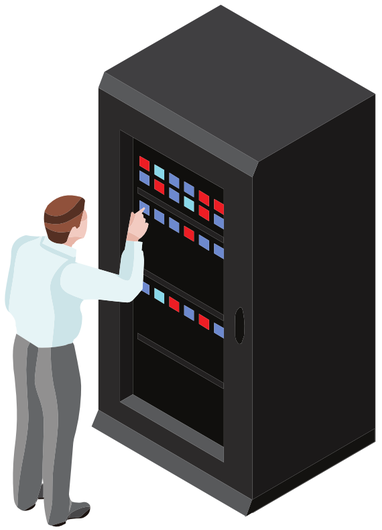


॥ सिद्धिः भूषयते विद्याम् ॥

avantika UNIVERSITY

DESIGN PROJECT | USER CENTRED DESIGN

TERMINAL APP

NIPUN PATEL
AU19B1009

Index

Contents **Page Nos.**

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.

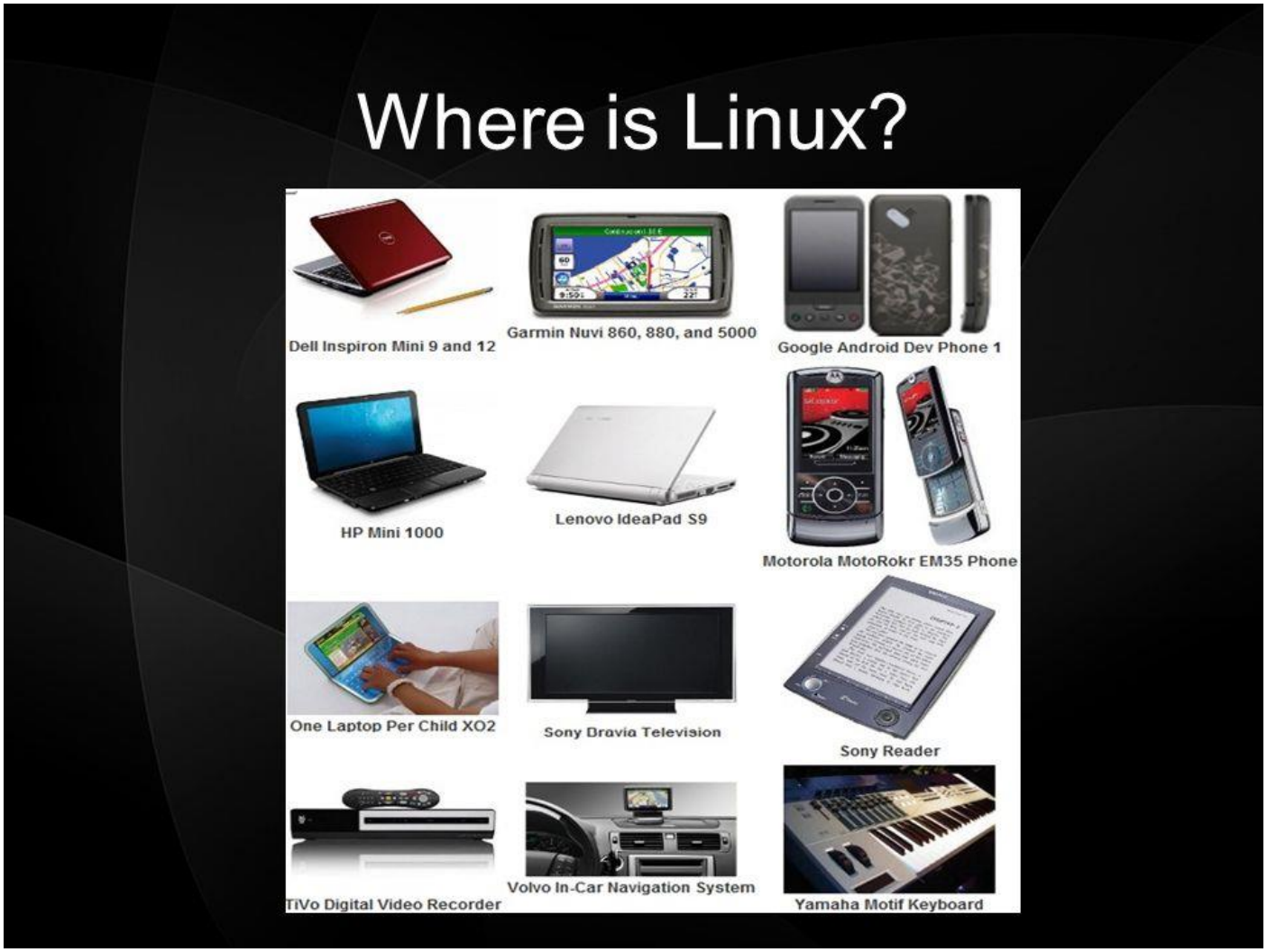
System user space
The administrative layer for system-level tasks like configuration and software install. This includes the shell, or command line, daemons, processes that run in the background, and the desktop environment.

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

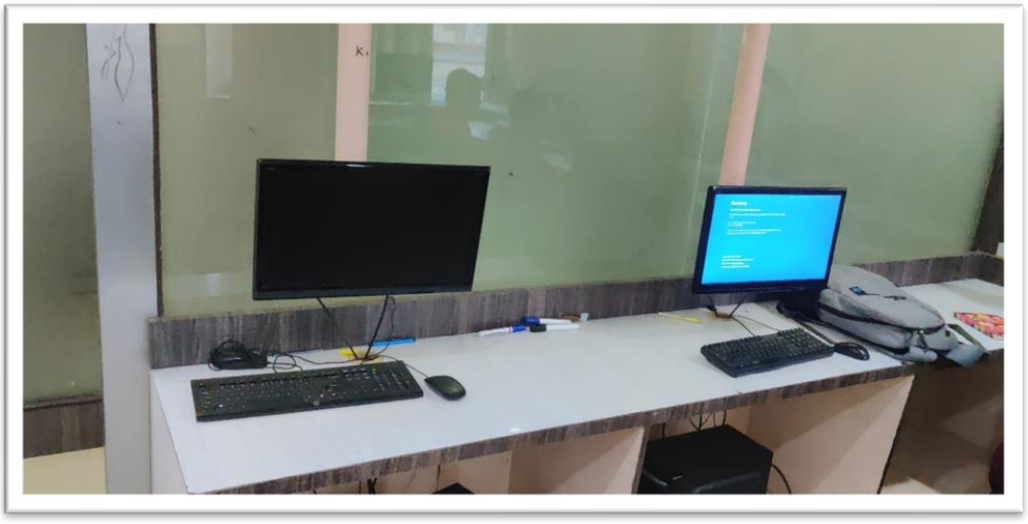
Secondary Research:

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

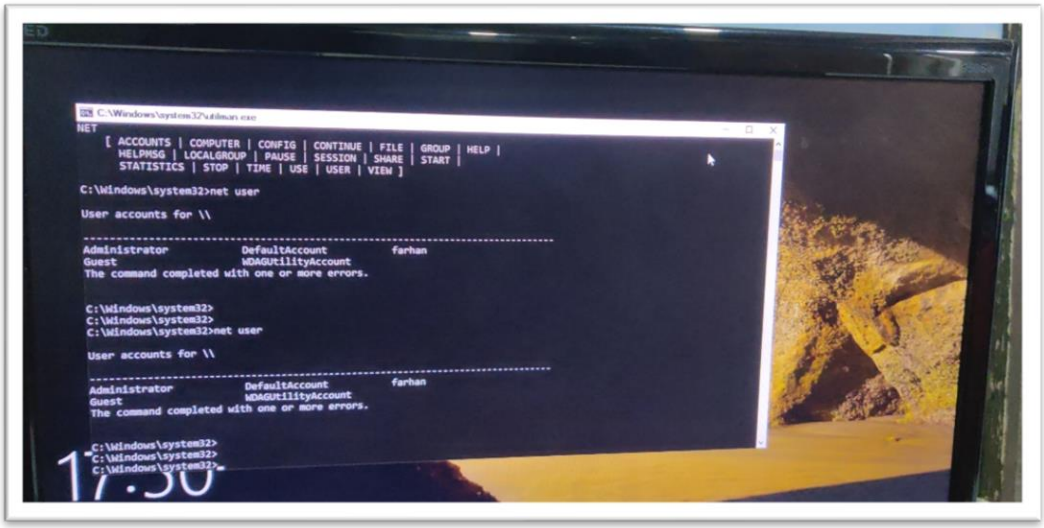
❖ Origin of the research work:

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



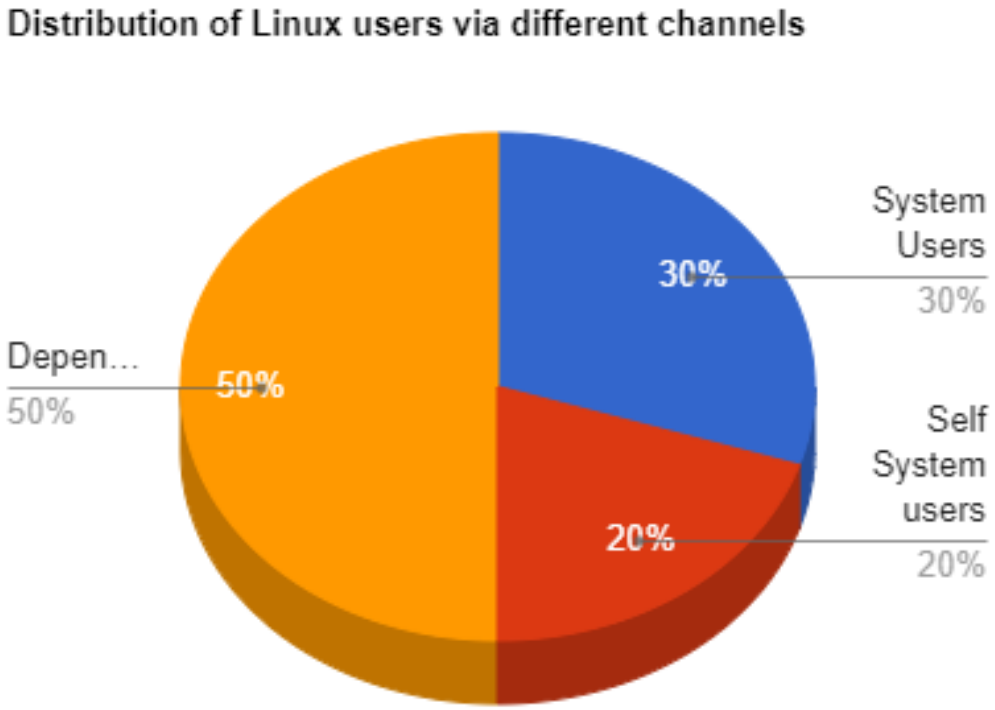
GUI based PC's runs virtual machine for Linux



Creating an Apache Server on Red hat RHEL 8.0

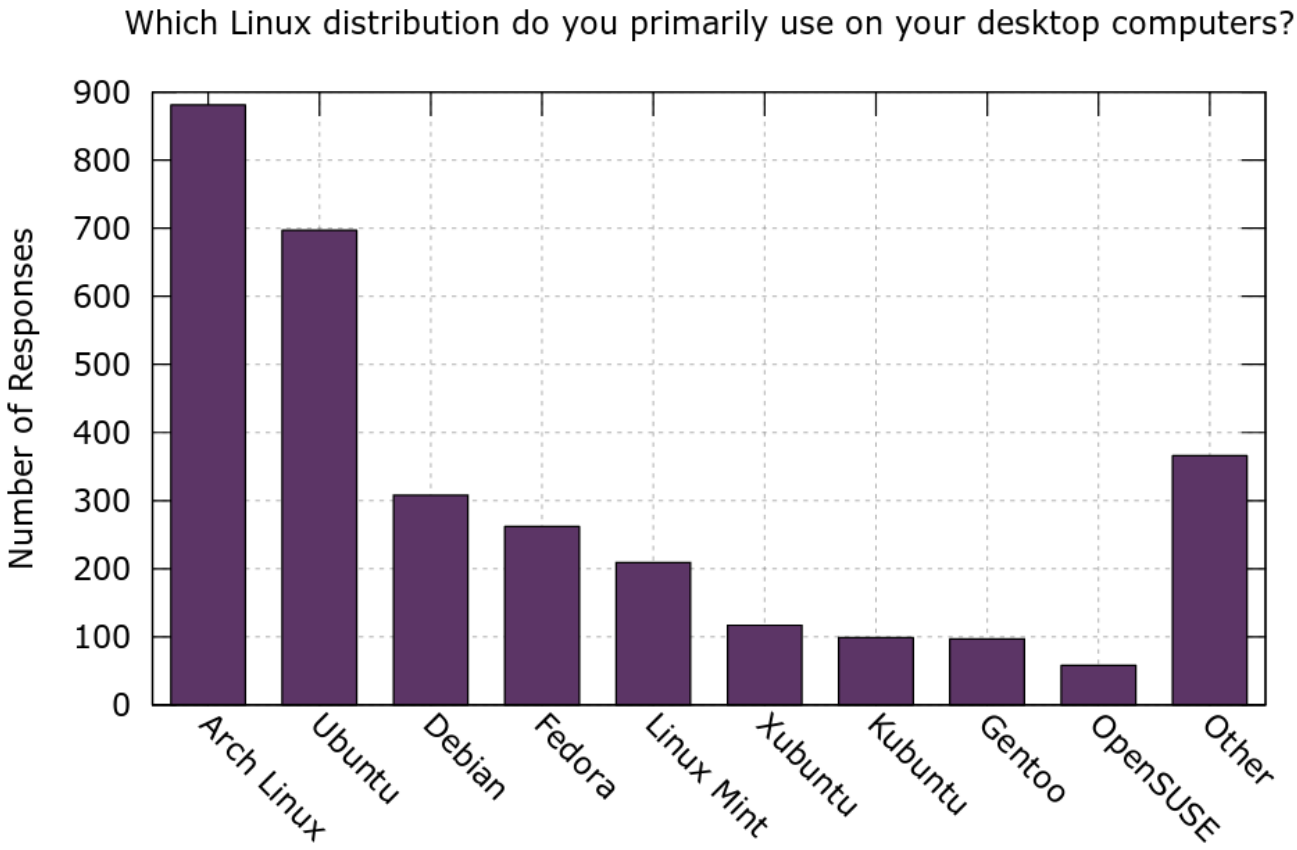
Interaction with Instructor Mr. Tausif Khan Sir

❖ Statistics:



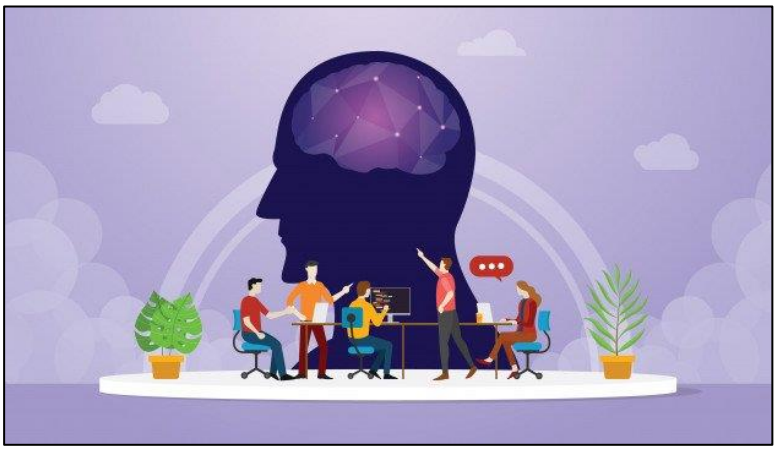
Distribution of users in IANT using Linux via different channels

Top Organizations uses Linux operated systems for their various services.



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.

FUN WITH LINUX

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

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.)

MIND MAP

IDEAS

1

Creating a mobile application for the user to interact with redhat linux terminal

- *creating a login/signup for with terminal credentials*
- *Giving test benefits by giving acces to mock tests*
- *creating a login/signup for with terminal credentials*

2

Creating a web interface that can be accessed publically from the hosted server, it will act as an online Linux terminal compiler which will be easy to use and convenient to access

- *web admin panel is also required for security access*
- *we need to buy domain or we can host on AWS*

3

We can connect local systems to the server system, and each local system can be shared over the internet which can be accessed to a particular IP via server API and can resolve the problem of user load

- *Multi accessibility on portal as well as system for same user*
- *It may create an ambiguity of user access at the same time 2 places*
- *Handling multiple things may degrade performance and system architecture*

4

- Dedicated storage to every registered user by applying ACLs to each user and adding file quota and Inode
- Setting permissions for providing a certain file type only to the server disk
- Handling user traffic by load balancing if possible on the server or we can freeze the access for particular time period

5

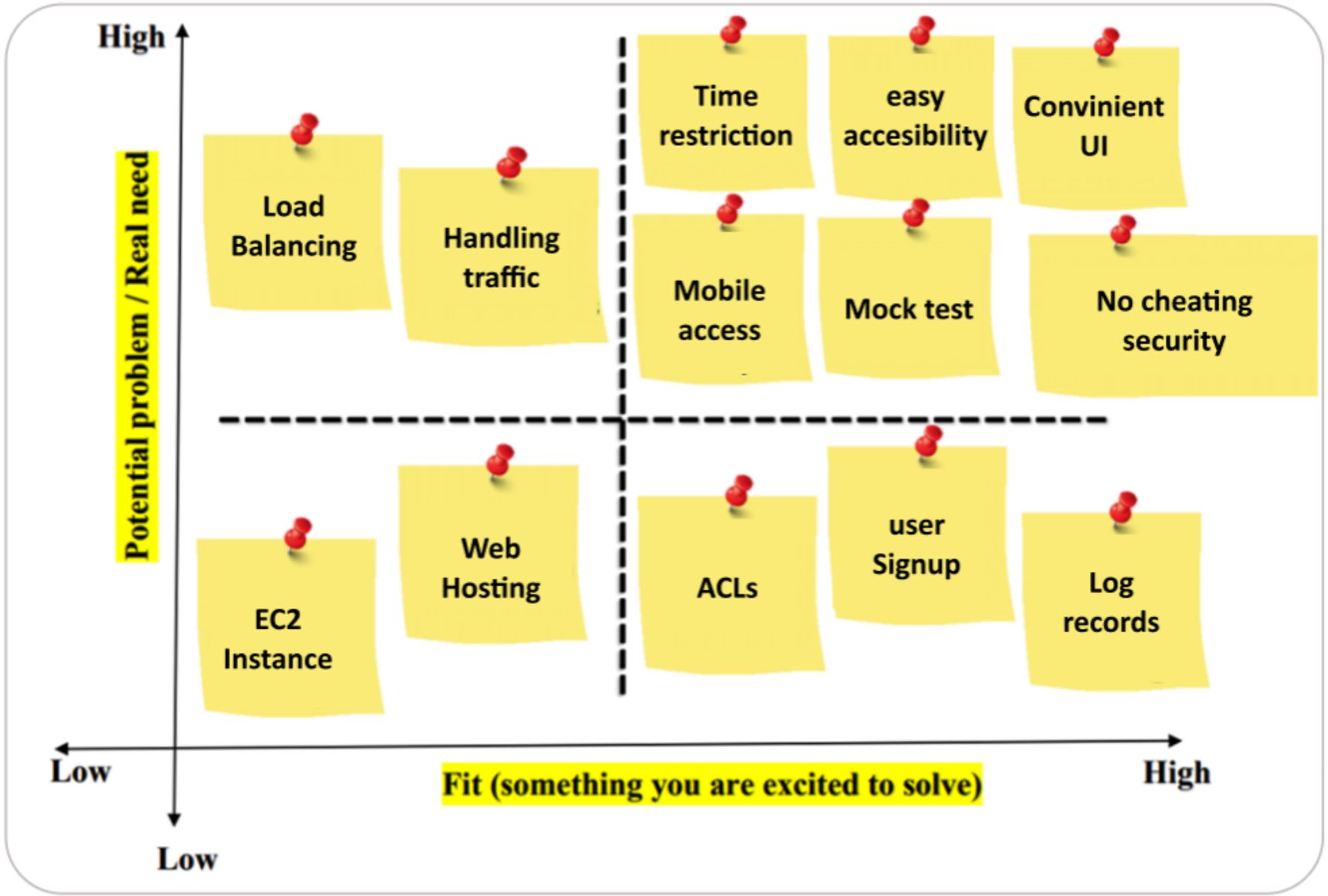
Instead of hosting our own website, we can train our users to use AWS EC2 instance and create Linux machine over there and practice on amazon's machine with the RedHat commands

- *High speed internet connectivity will be a boost to performance*
- *Multiple machine can be created on for single users*

- Note: In the above mindmap all the ideas are randomly generated and can have implications to predict but can plunder the system as well meanwhile with the ideas questions and pain points are raised wherever required in the relations created

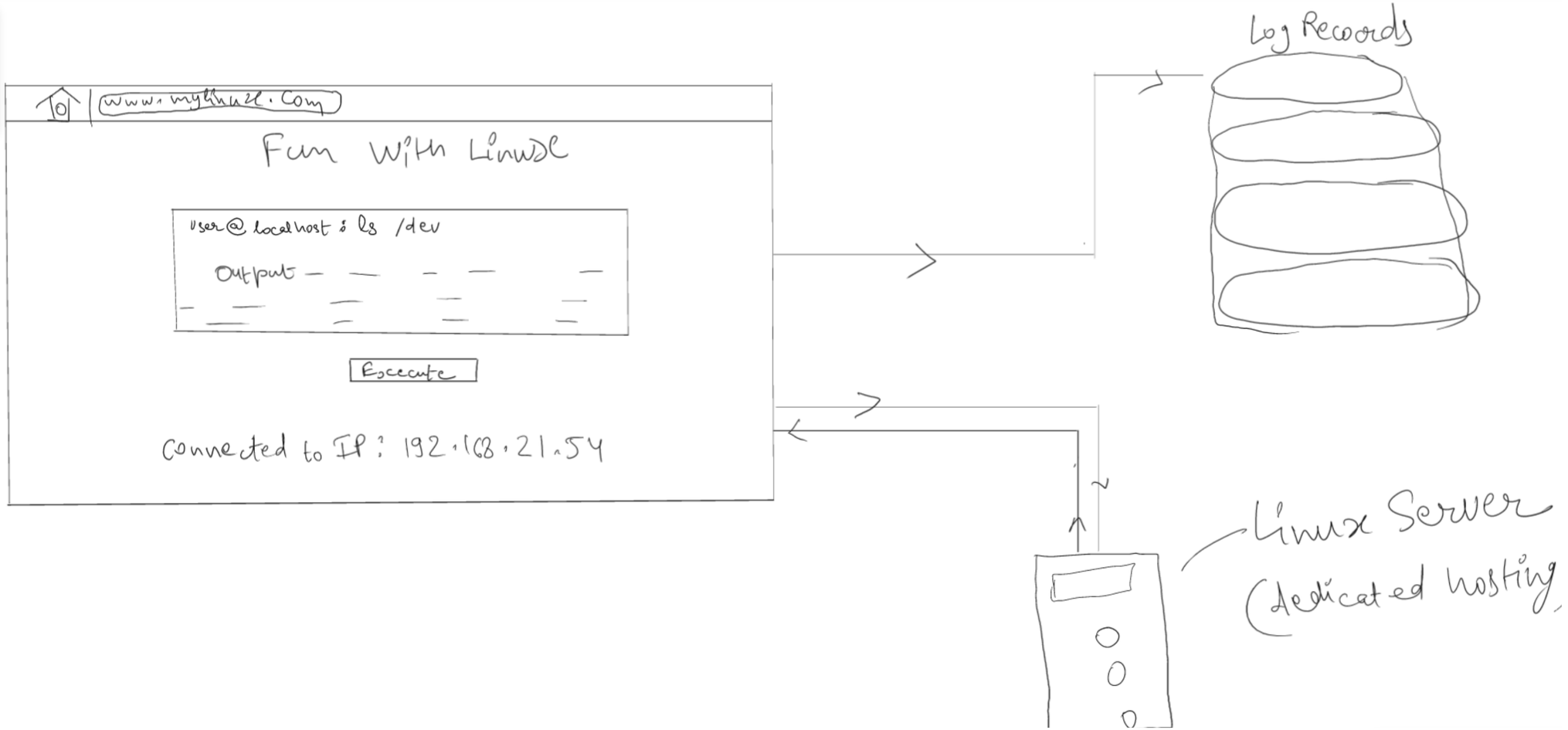
KANO MODEL ANALYSIS

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

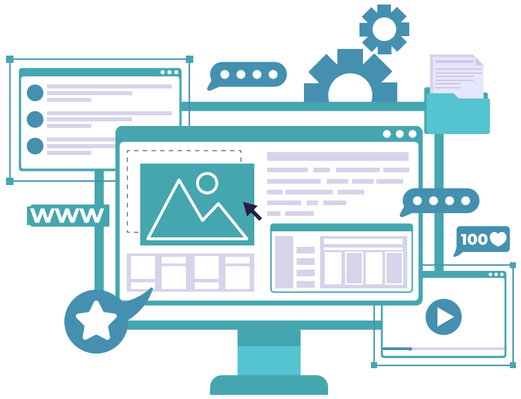


SKETCHES AND CONCEPTS

01



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

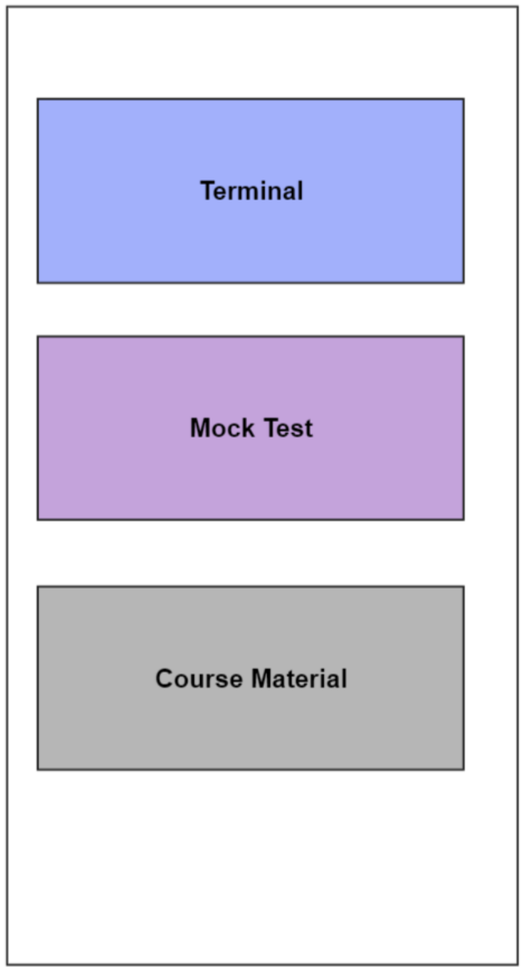
- On our server, dedicated self-hosting makes access simple.
- Convenient UI for easier understandability of terminal
- Direct access to the compiler saves the time of the user
- A high-performance server system and suitable high-speed internet are an ideal match.
- Realtime accessibility on any device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

CONS

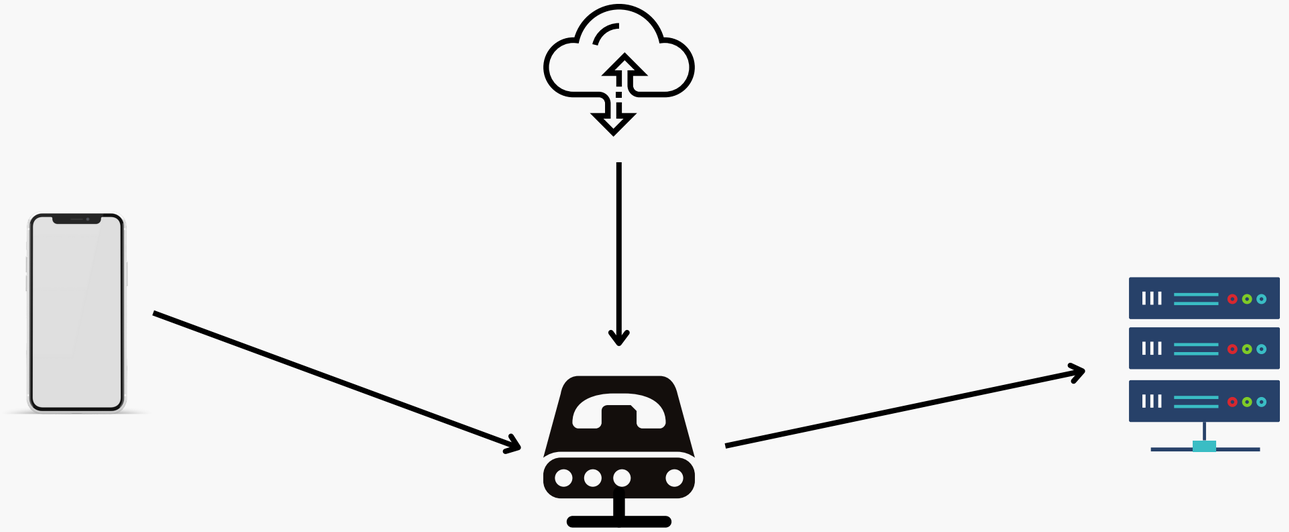
- Fear of getting hacked as it is web-hosted
- No test practice session is available for use at the terminal
- No Proctoring or firewall to monitor the command activity may kill or halt the system
- The user traffic and accessing simultaneously may affect load balancing on the server
- Lack of features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

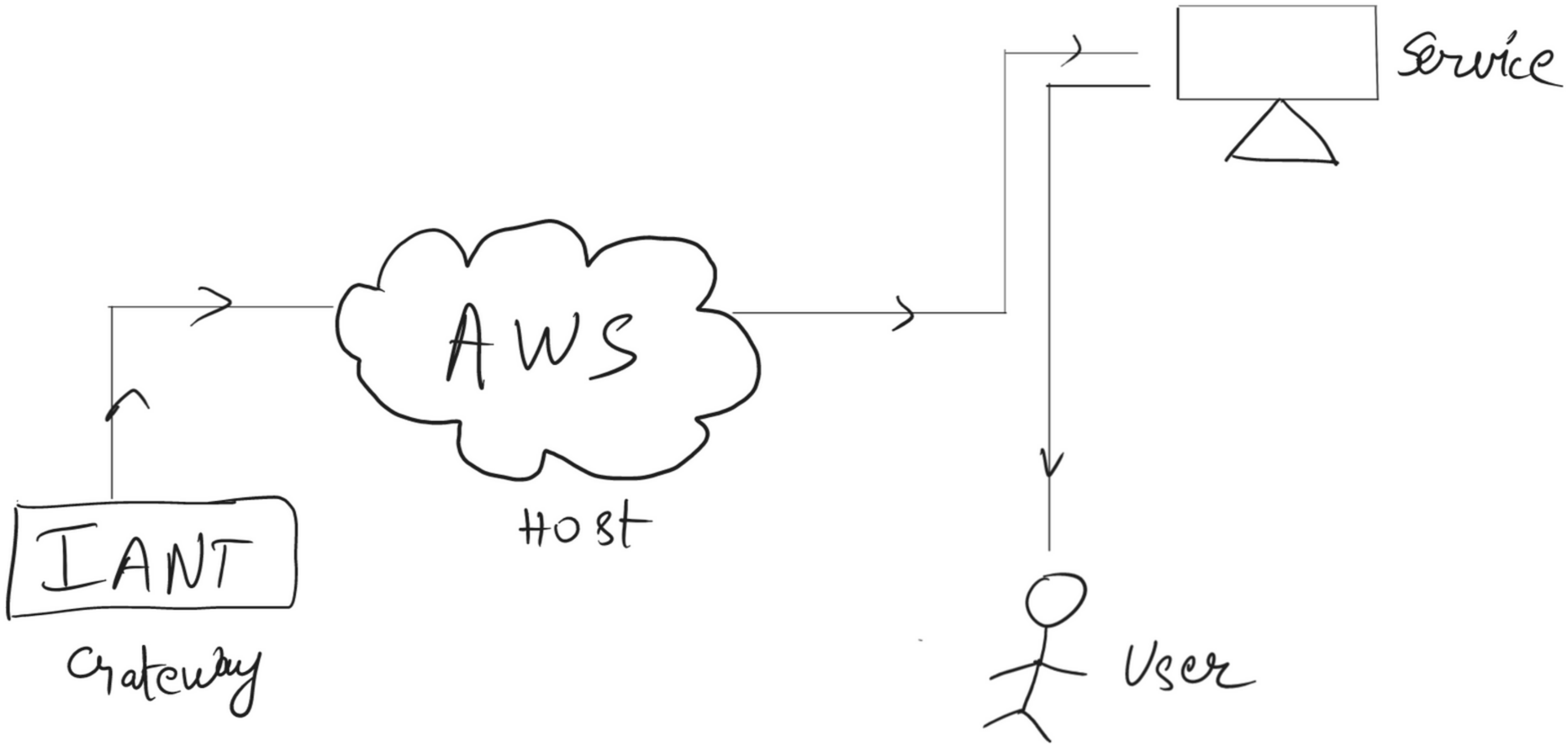
- Support multiple features which is a benefit for the user
- Convenient UI for easier understandability of terminal
- A cost-effective solution, easily accessible to everyone
- Private access with dedicated storage is a boom
- Dedicated institute support helps in growing the sector
- Improves performance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

CONS

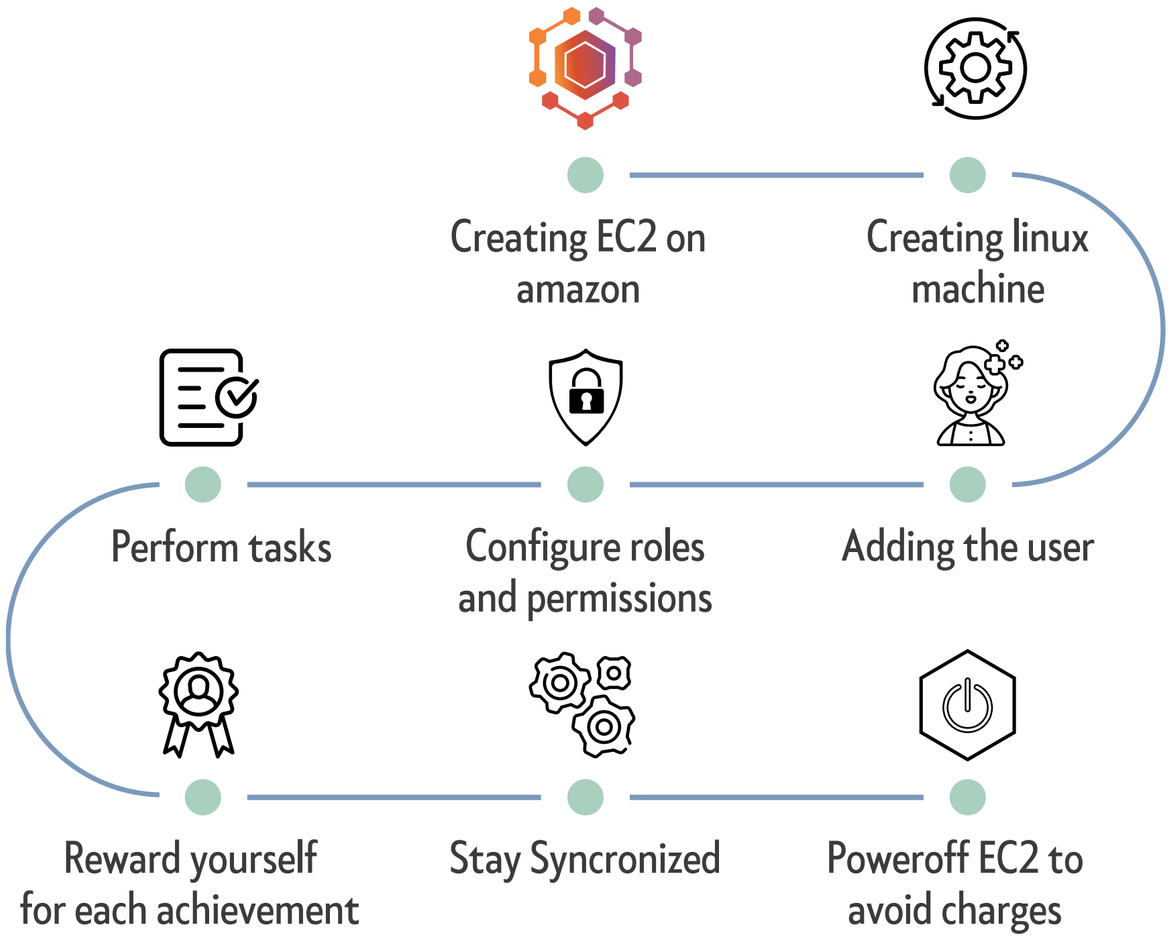
- Avoiding cheating is a concern of doubt
- Limited screen size can clutter the workspace
- Scalability and load balancing is a concern

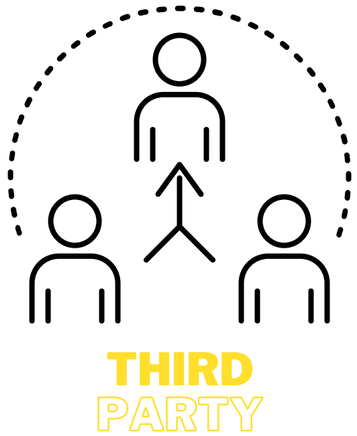
Cloud Service

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

Comparison Analysis Finalization

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

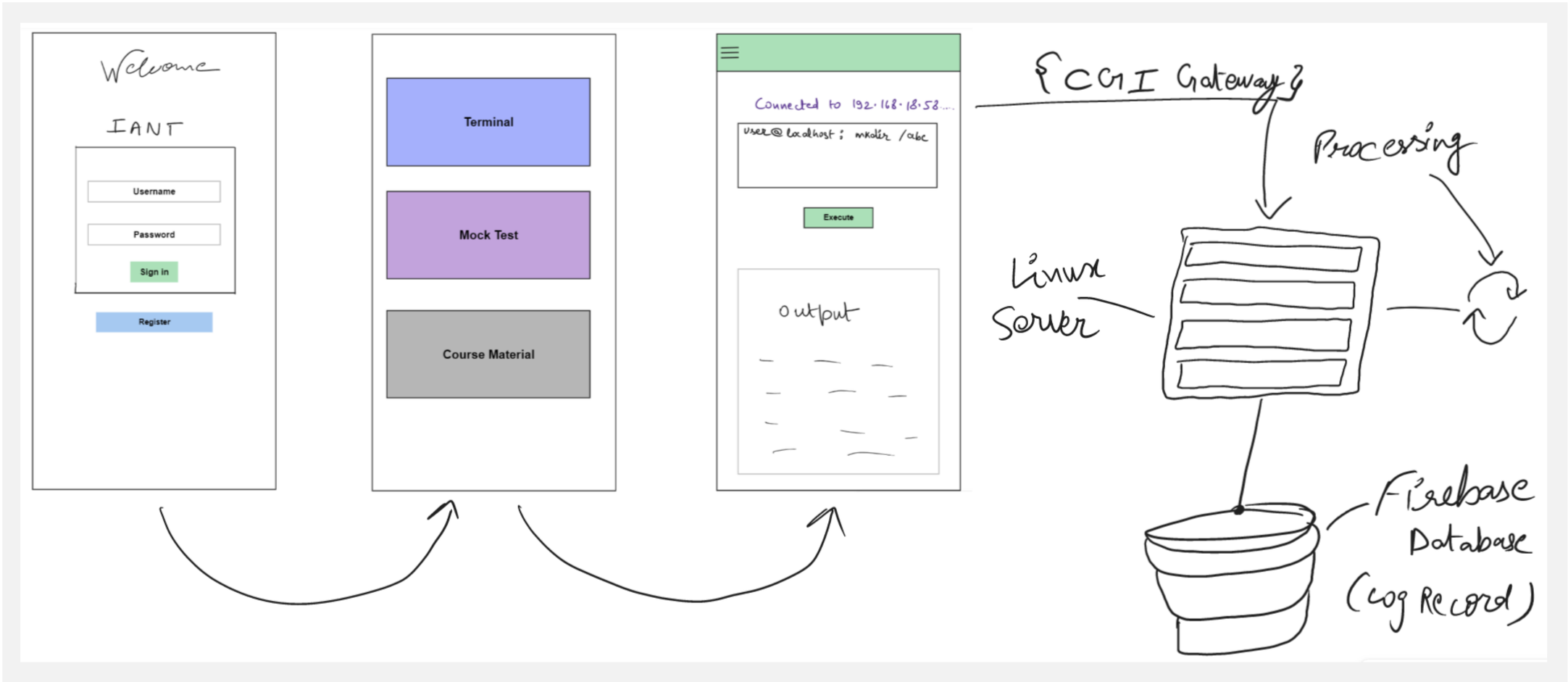
Impact

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

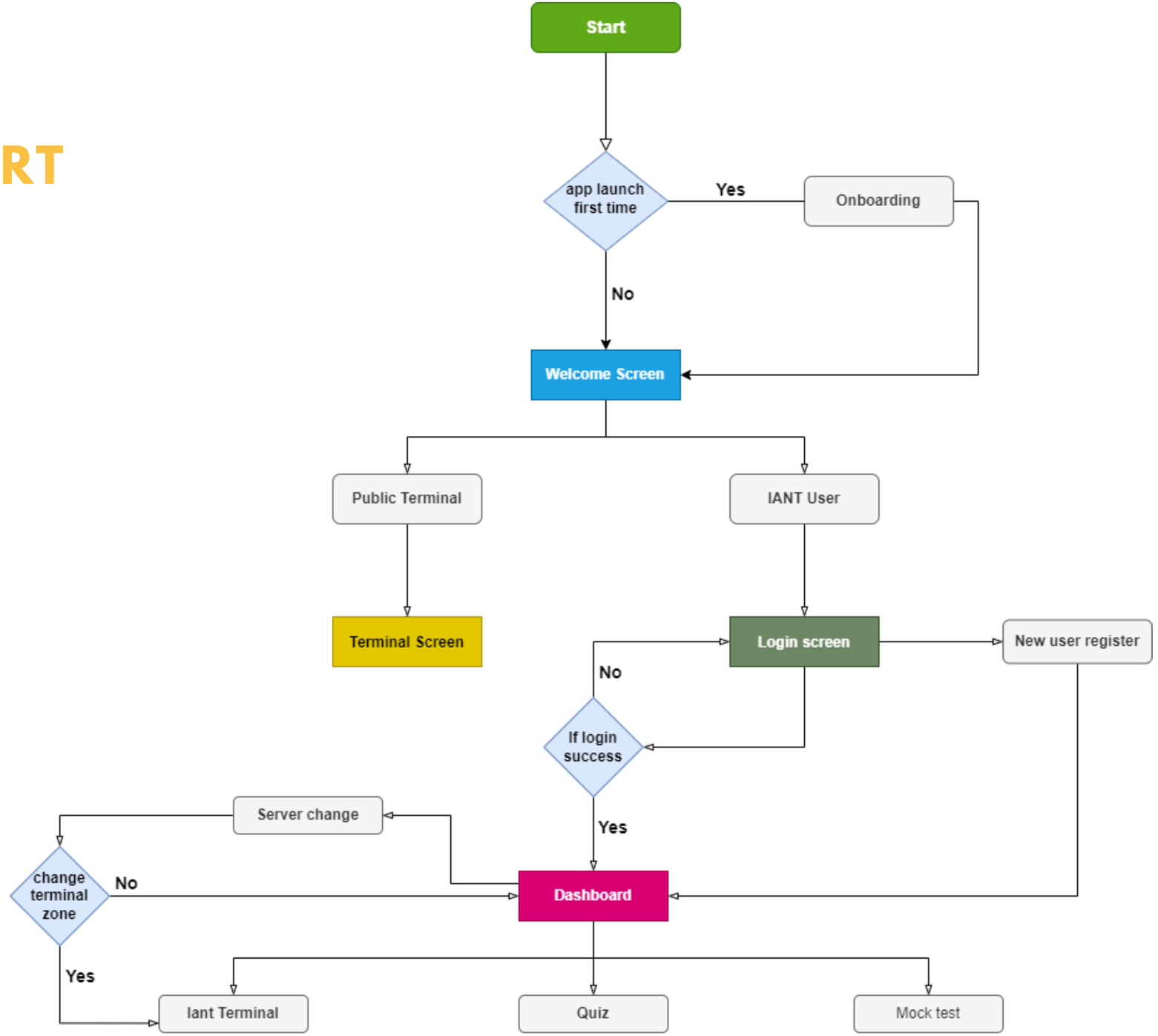
Admin can apply dedicated permissions to each registered user

STAGE 3

DESIGN

40% on progress...

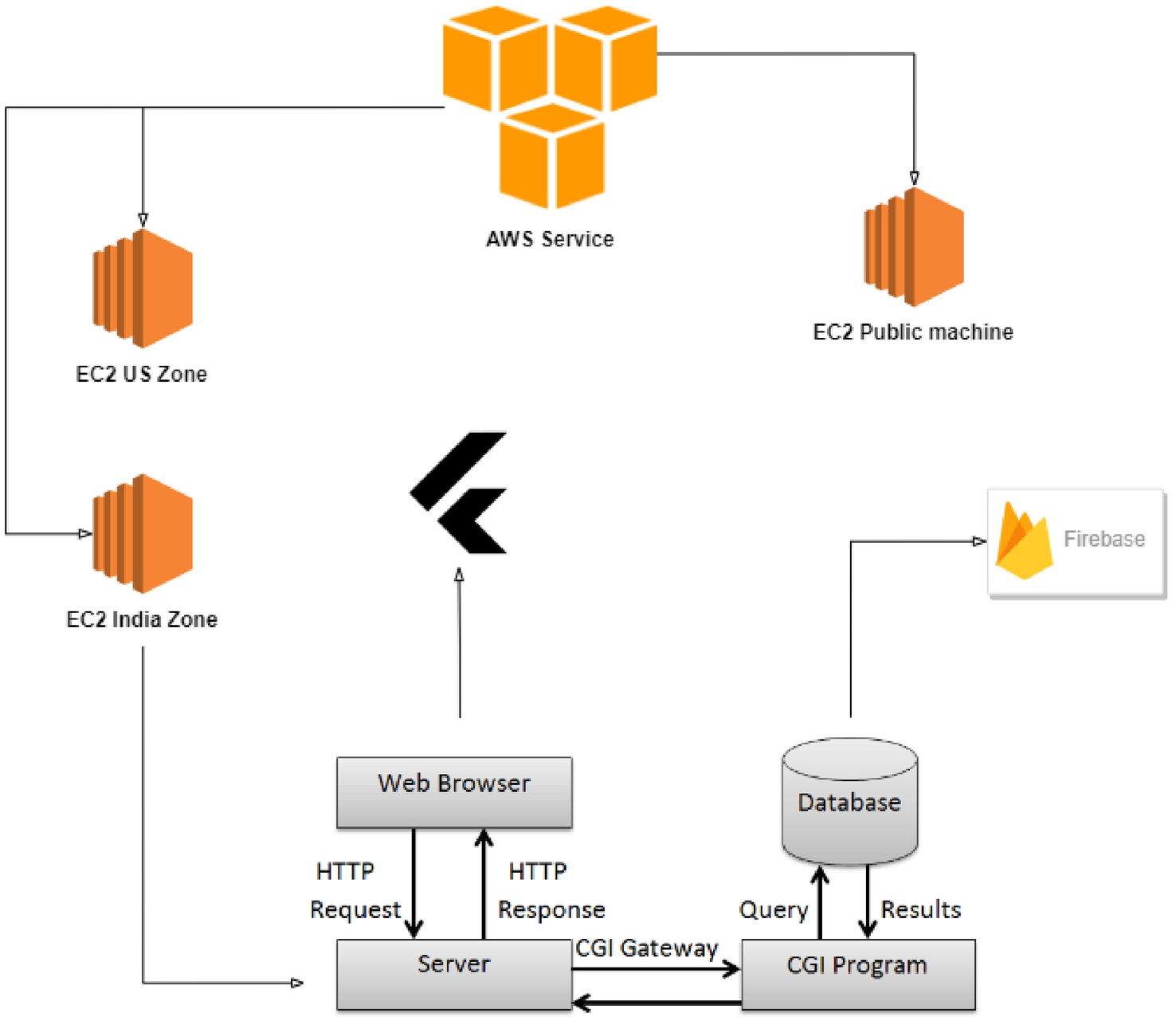
FLOWCHART


Start
app launch first time
Yes
Onboarding
No
Welcome Screen
Public Terminal
IANT User
Terminal Screen
Login screen
New user register
No
If login success
Server change
Yes
change terminal zone
No
Dashboard
Yes
Iant Terminal
Quiz
Mock test

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

Request handling

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

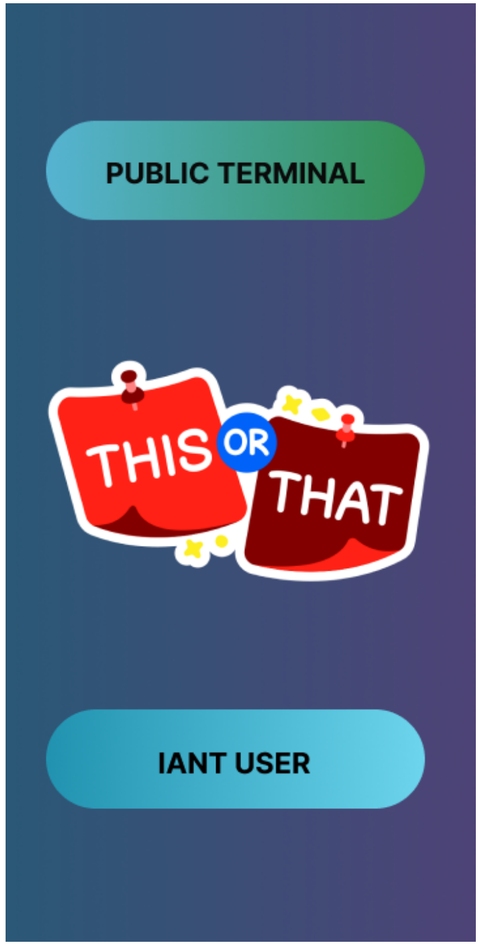
Quiz practices

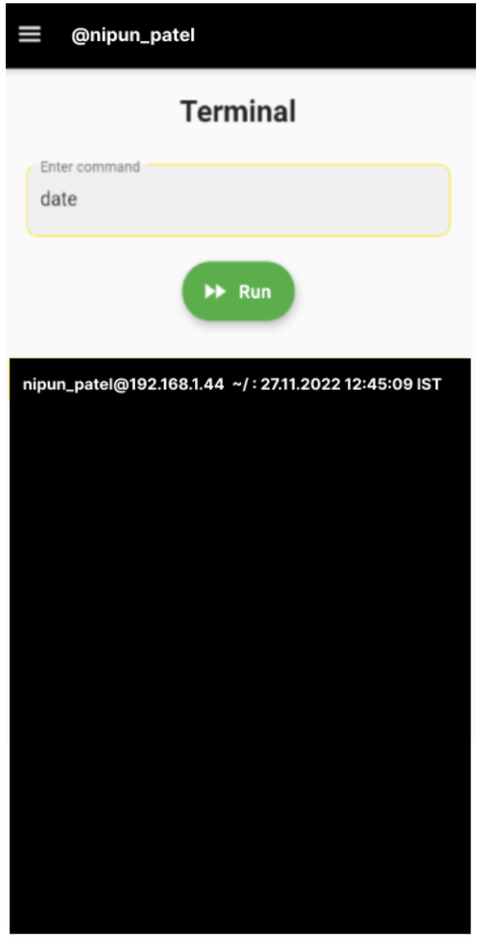
Time-based control in each quiz must be a fair delivery

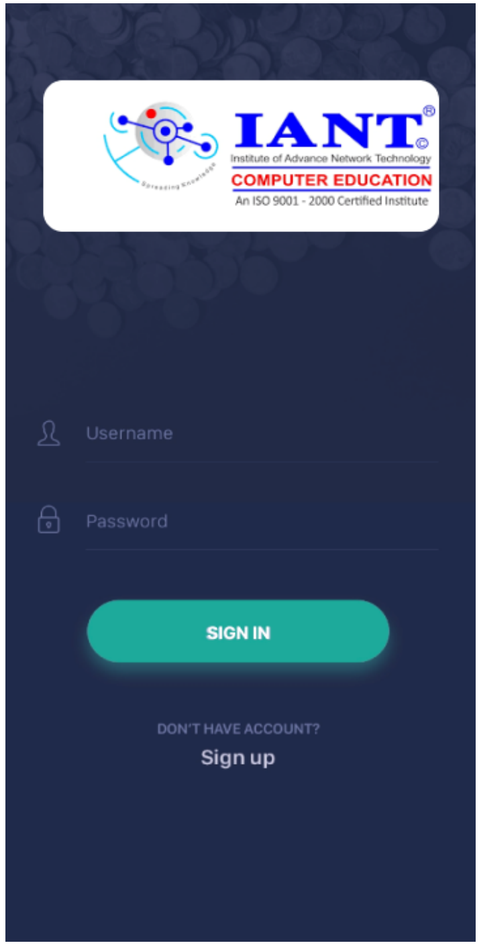
EC2 Scripting

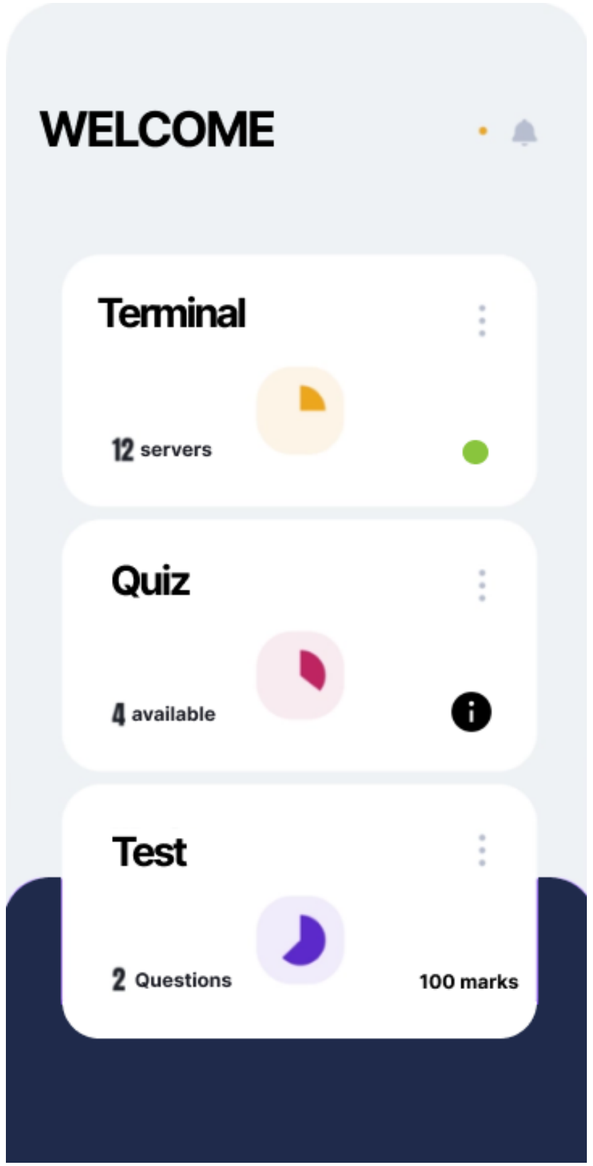
AWS cloud-hosted machine requires configuration which must be correct

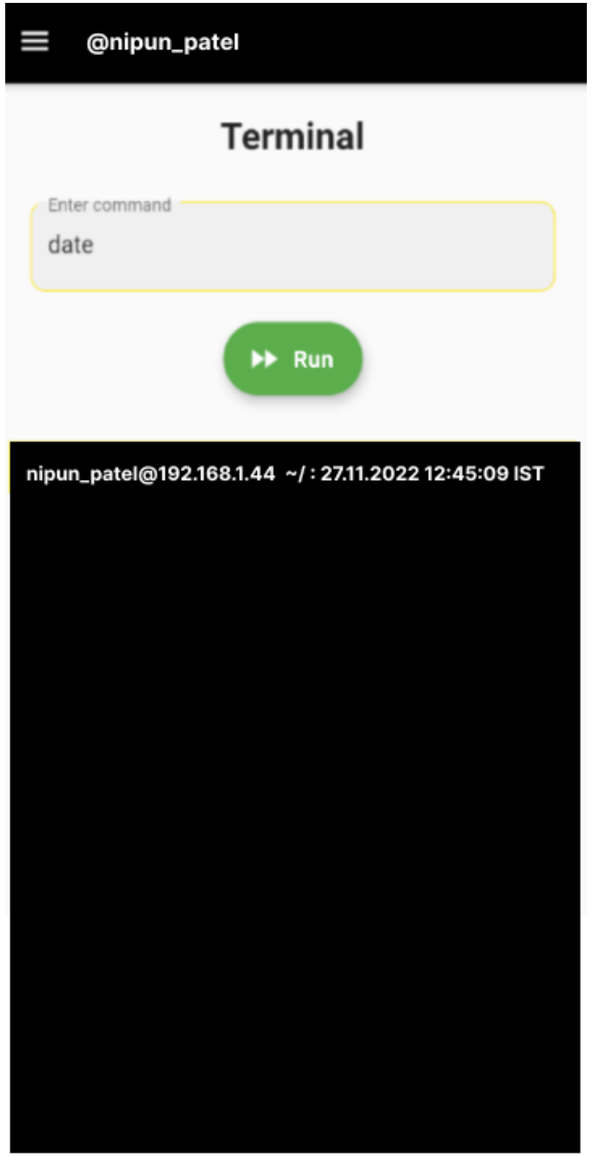
WIREFRAMES

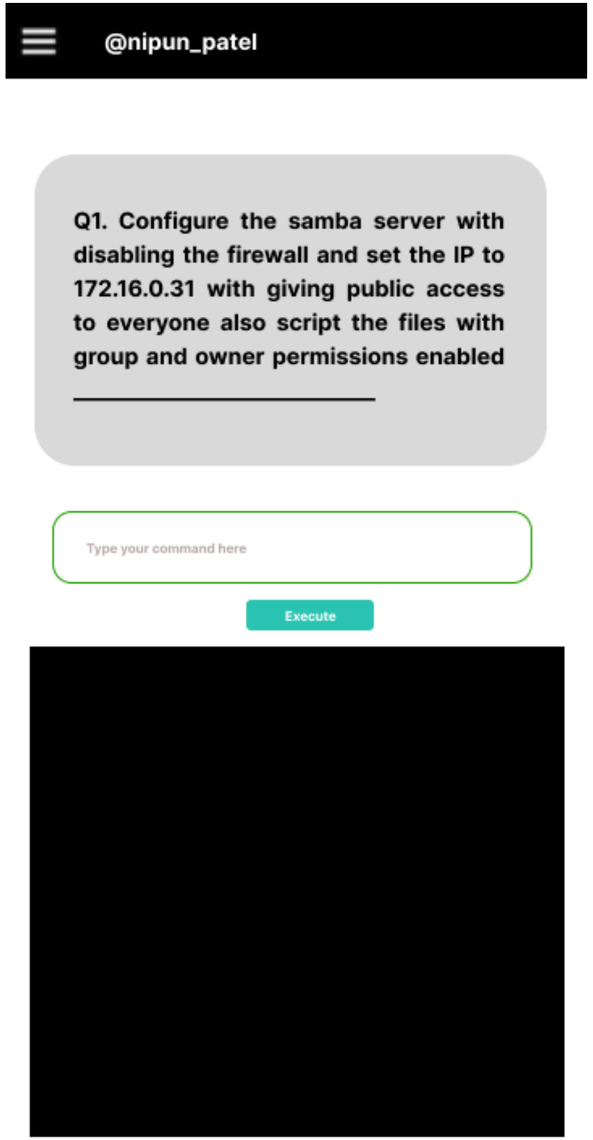

PUBLIC TERMINAL
THIS OR THAT
IANT USER

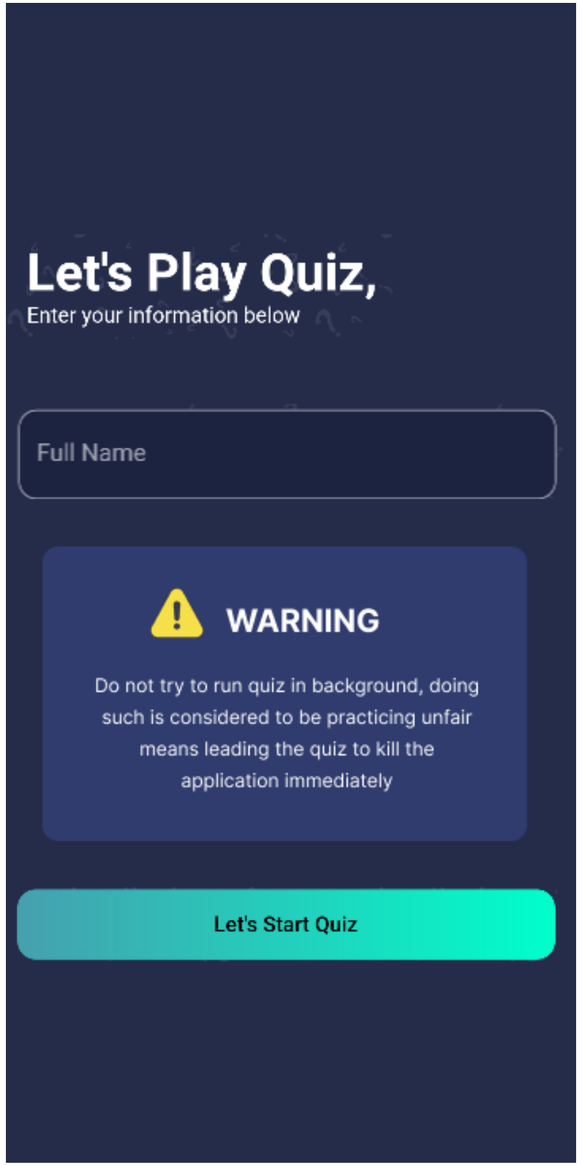

@nipun_patel
Terminal
Enter command
date
Run
nipun_patel@192.168.1.44 ~/ : 27.11.2022 12:45:09 IST

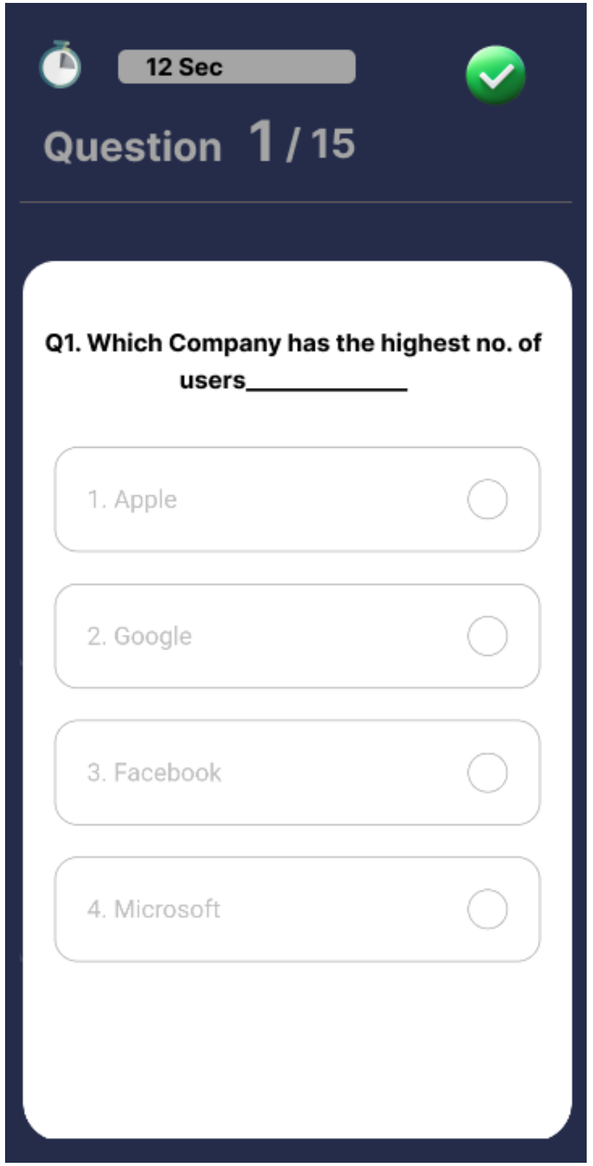

IANT
Institute of Advance Network Technology
COMPUTER EDUCATION
An ISO 9001 - 2000 Certified Institute
Username
Password
SIGN IN
DON'T HAVE ACCOUNT?
Sign up

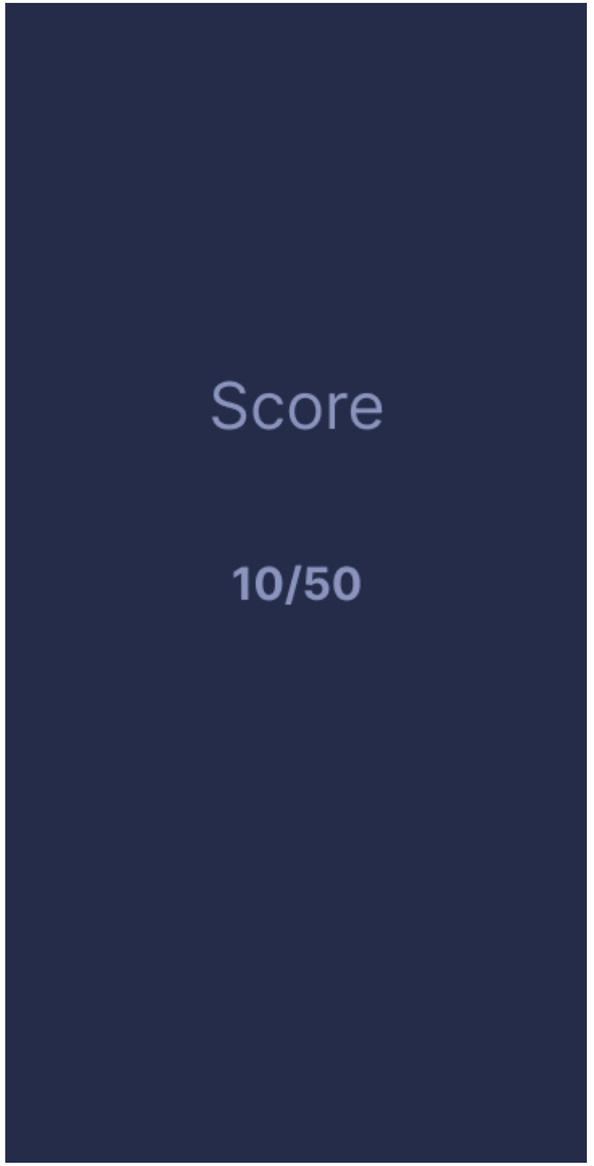

WELCOME
Terminal
12 servers
Quiz
4 available
Test
2 Questions
100 marks


@nipun_patel
Terminal
Enter command
date
Run
nipun_patel@192.168.1.44 ~/ : 27.11.2022 12:45:09 IST


@nipun_patel
Q1. Configure the samba server with disabling the firewall and set the IP to 172.16.0.31 with giving public access to everyone also script the files with group and owner permissions enabled
Type your command here
Execute


Let's Play Quiz,
Enter your information below
Full Name
WARNING
Do not try to run quiz in background, doing such is considered to be practicing unfair means leading the quiz to kill the application immediately
Let's Start Quiz


12 Sec
Question 1 / 15
Q1. Which Company has the highest no. of users_____
1. Apple
2. Google
3. Facebook
4. Microsoft


Score
10/50

USER DEFINED STYLESHEET

Primary Color

Secondary Color

Button rounds

Montserrat **Heading** **30pt**

Inter Subheading 25 pt

Inter Normal text 12 pt

Icons used

Bold

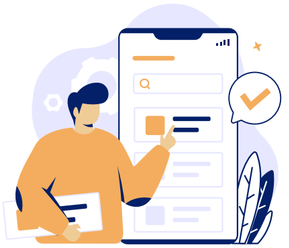
Italic

Underline

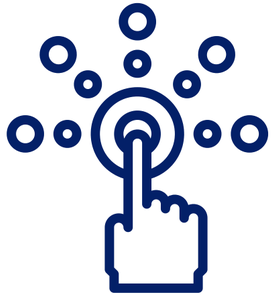
DESIGN FEATURES

Consistent UI

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



Interactive Animation



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

User defined colors

Color combination and company theme used in creating the wireframe

STAGE 4

IMPLEMENTATION

80% on progress...

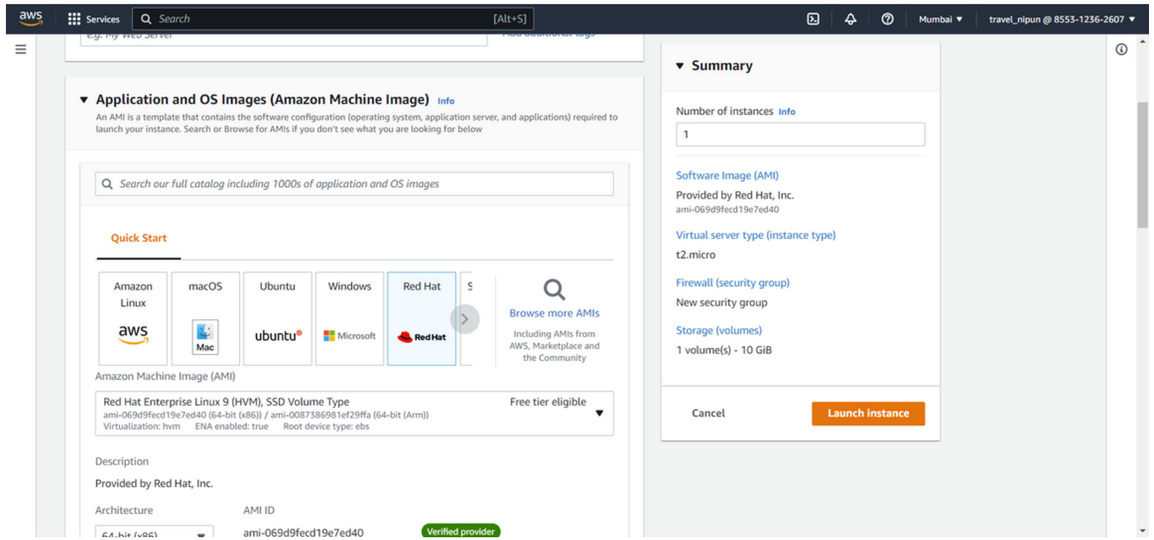
STAGES

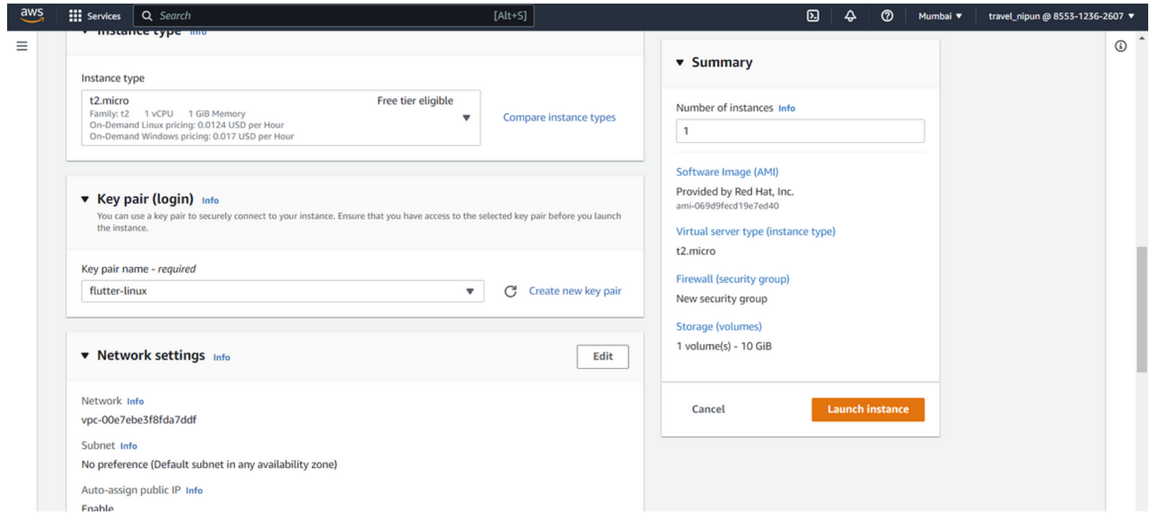
For the implementation phase, we have divided the procedure into three different stages that are setting the AWS environment, designing the backend, and connecting frontend

SETTING UP AWS ENVIRONMENT

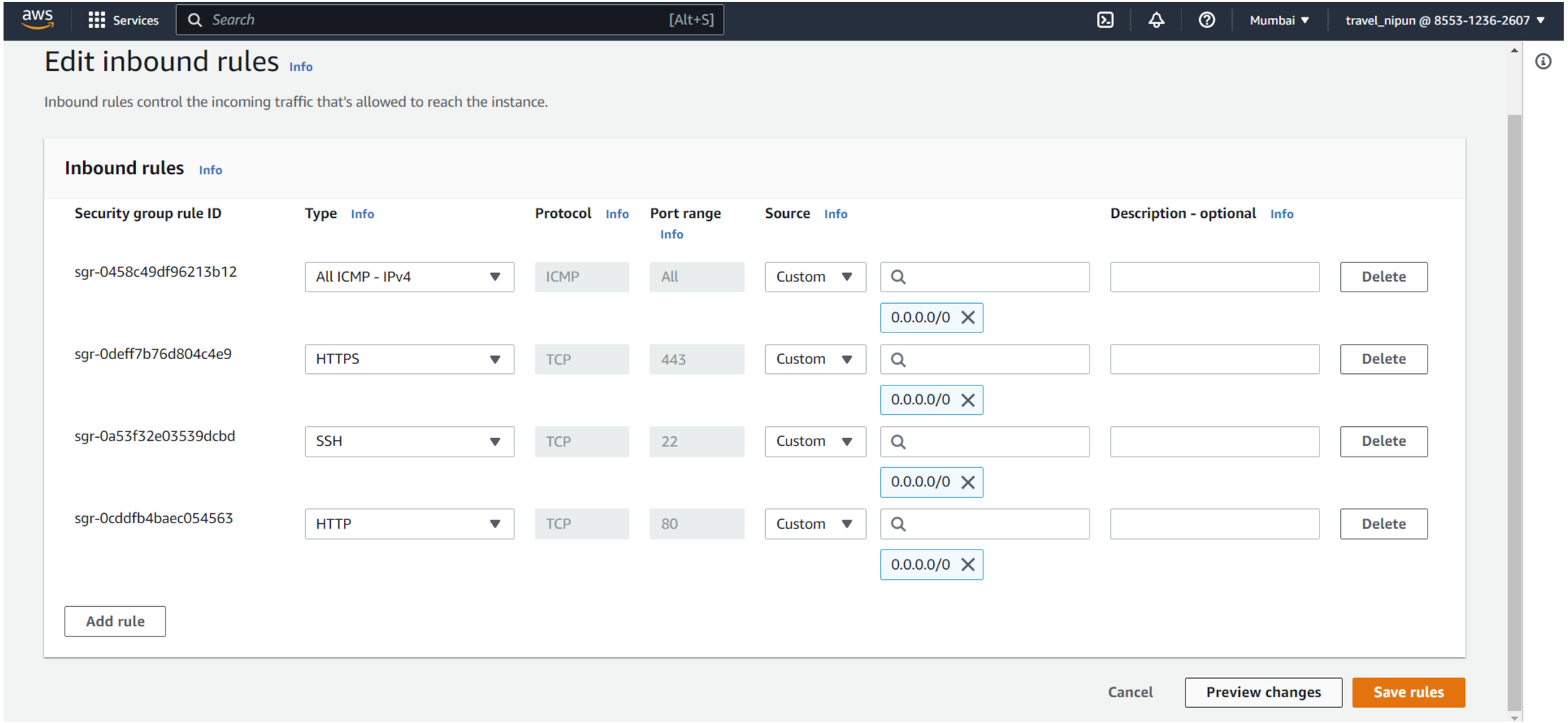
1

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.

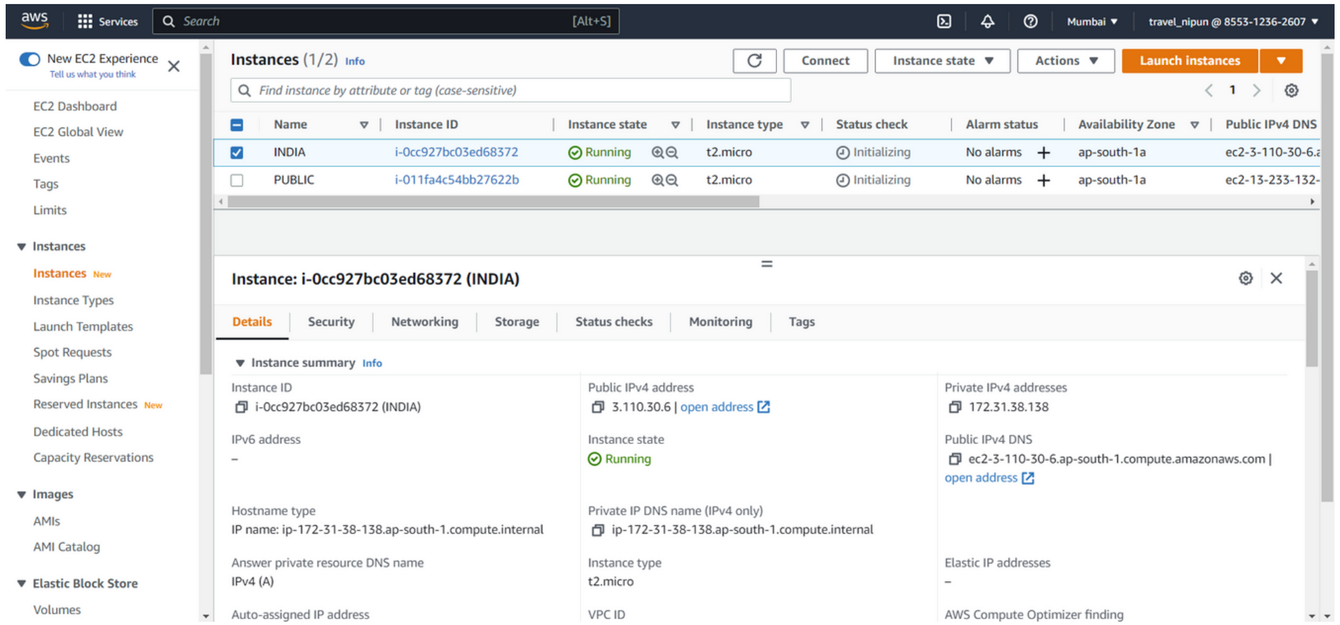


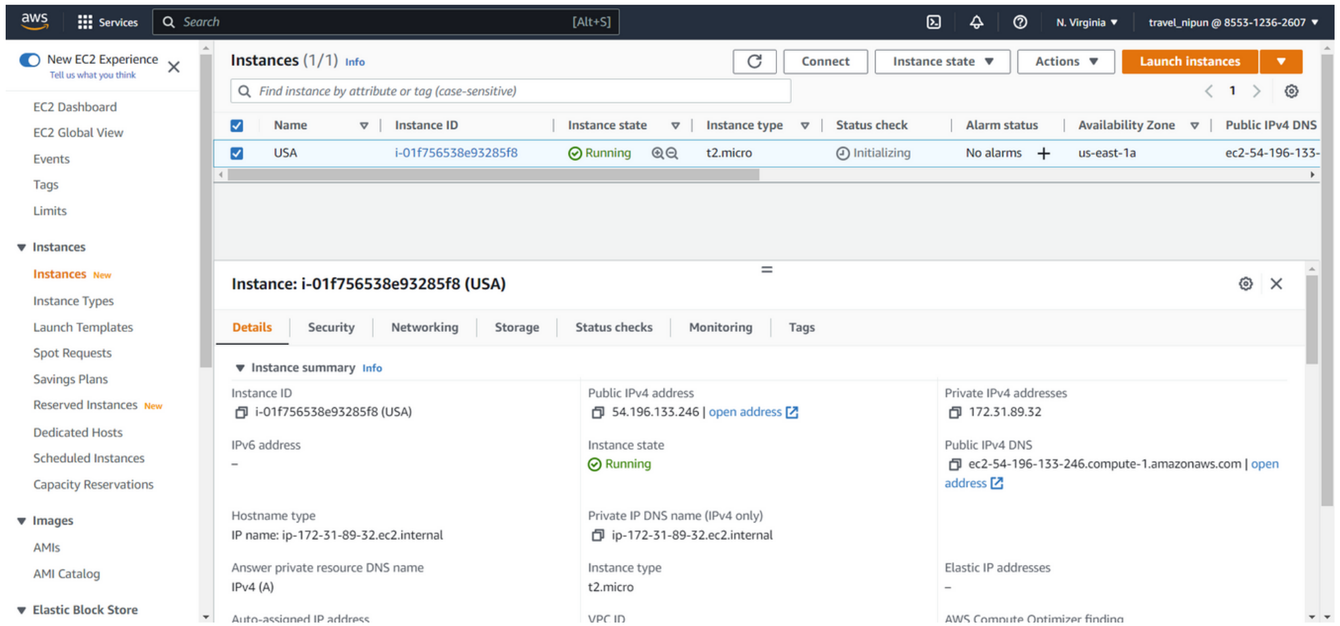


2. Select the key pair we have created so that the machine will log in with that security credentials, next select the security rules configured in the next step and hence launch the instance.

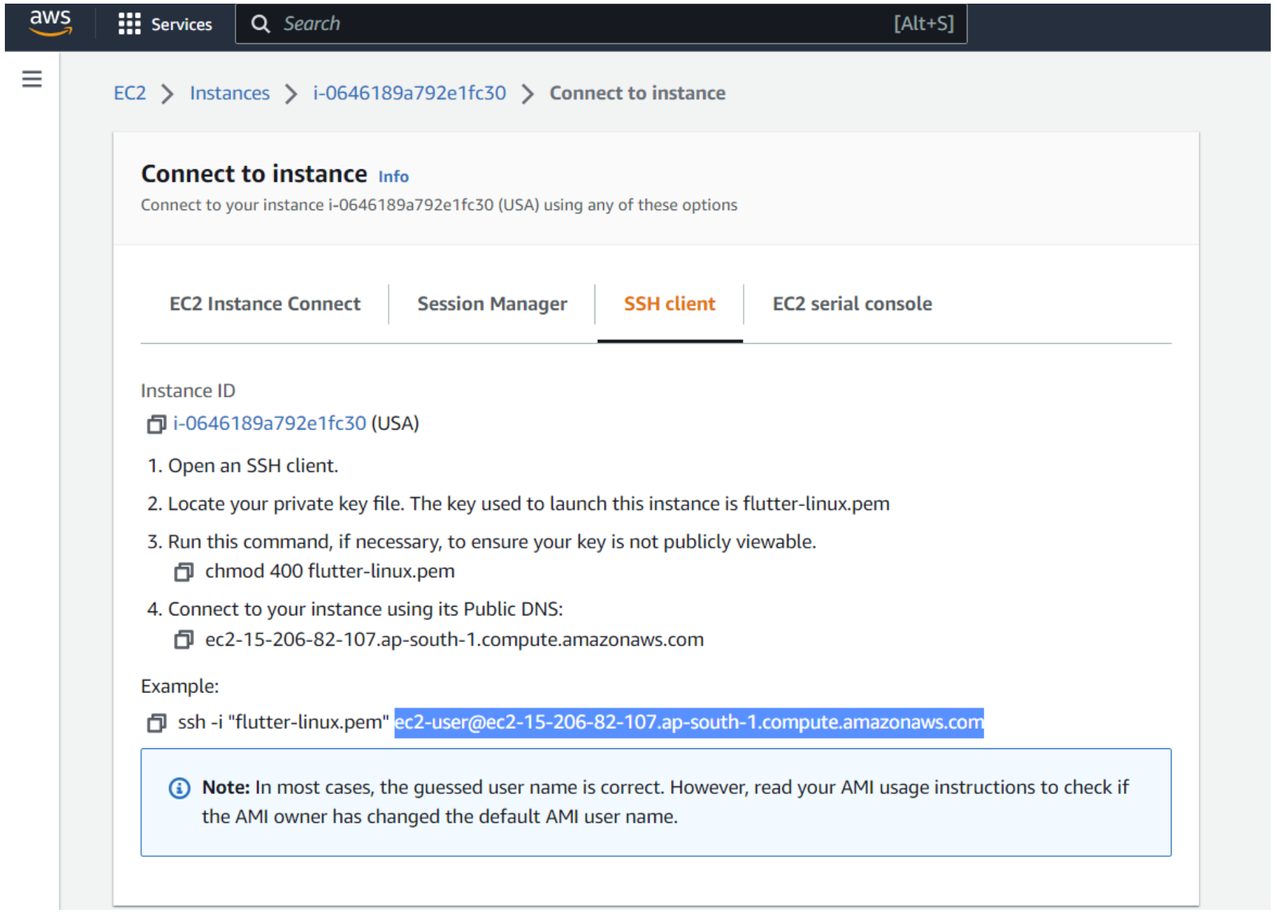


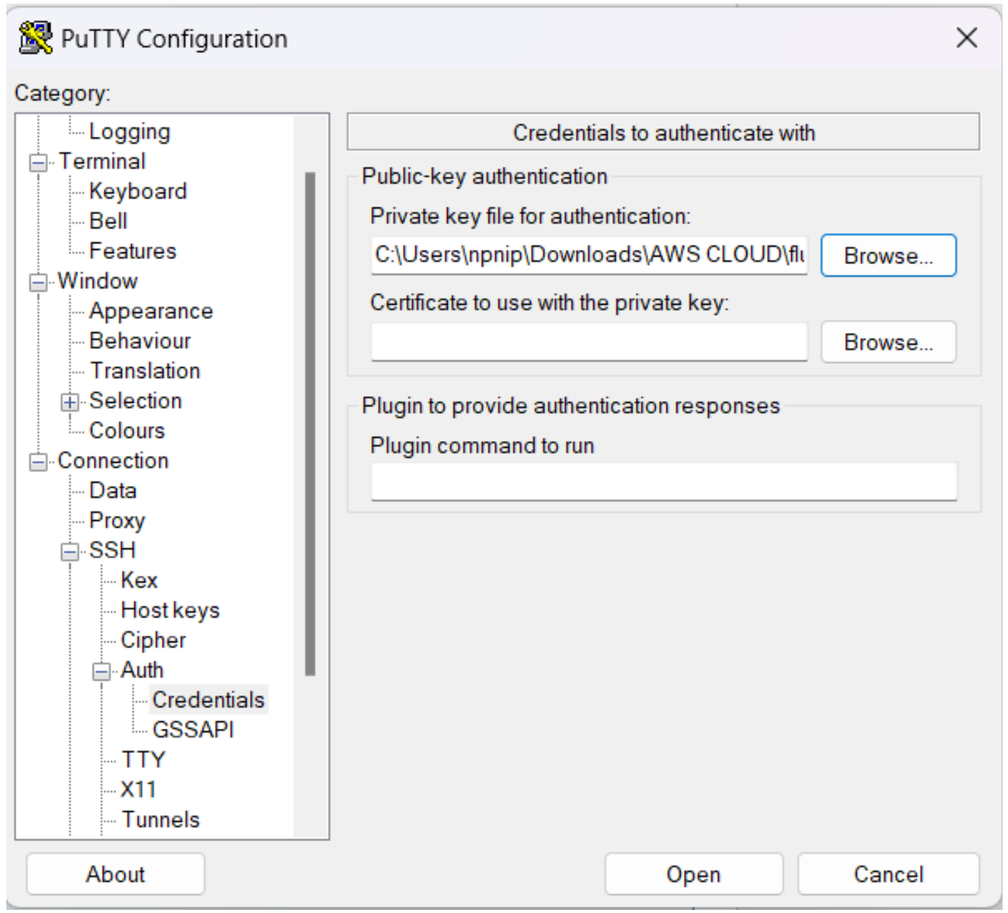
3. Go to the security groups we created and edit the inbound rules and give access to **HTTP, and HTTPS ports** as required for triggering the **web requests**. allow **ICMP ports** as they **send commands requests**, also allow **SSH Port** so that we can log in to the Linux machine with the private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





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.

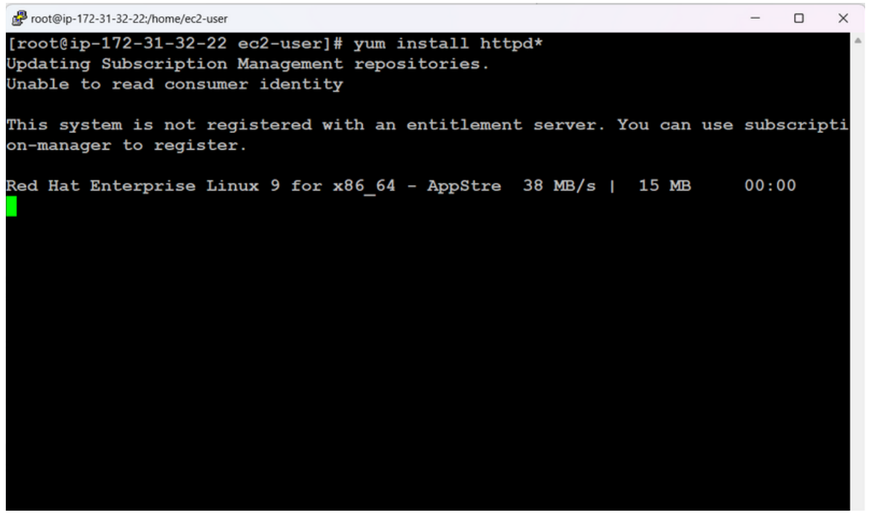


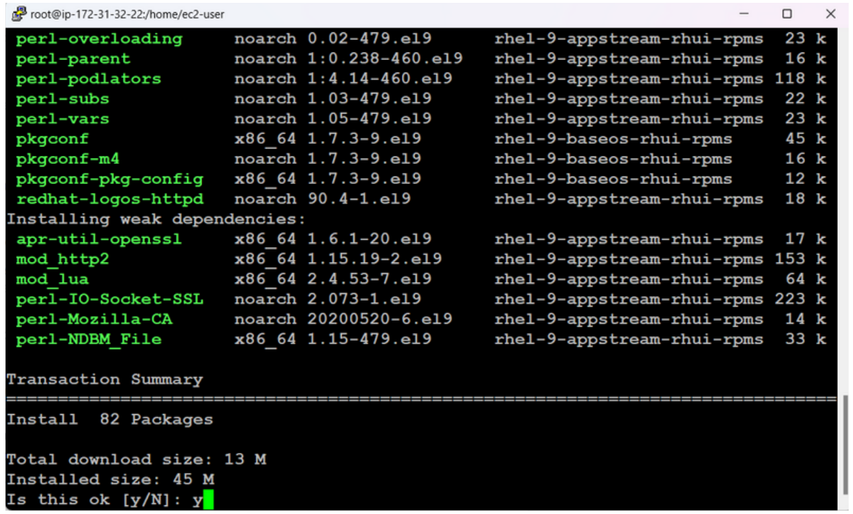


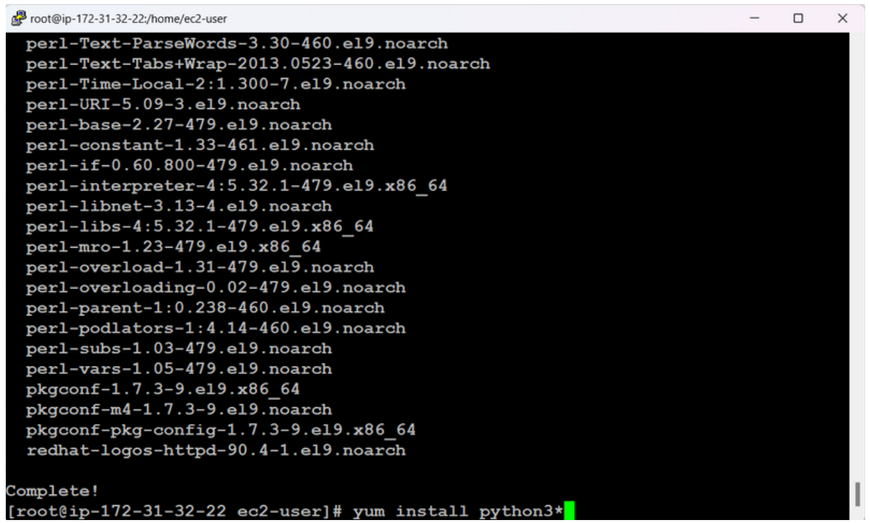
5. Now its time to configure our Linux server for that we need to connect to the machine using putty client, copy the ssh client URL and paste it the putty, now add the security key we have downloaded for putty it will come under **SSH---->AUTH---->Credentials** part, browse your key add it and connect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

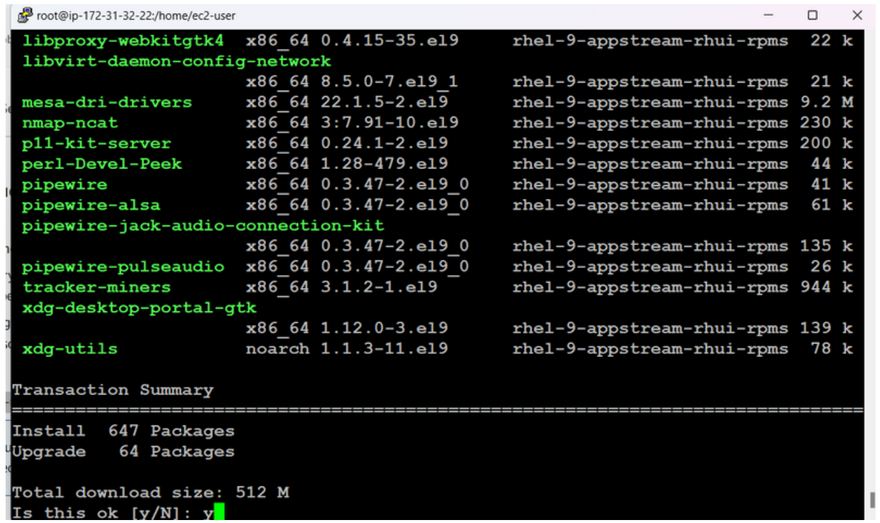
SERVER AND BACKEND CONFIGURATION

2







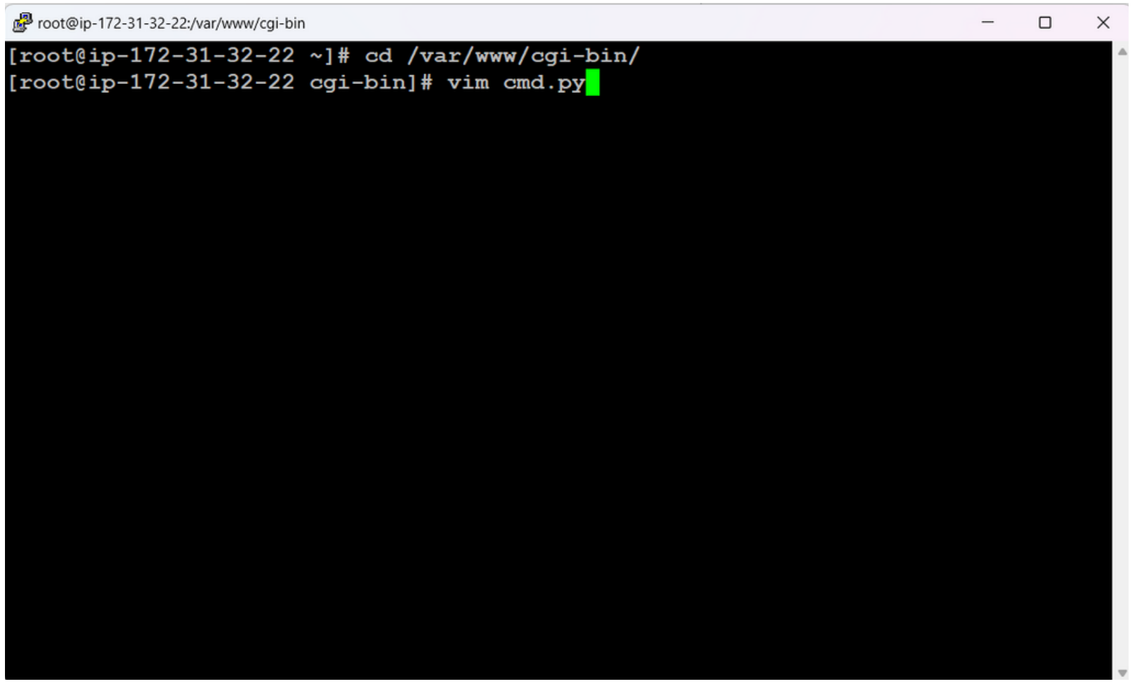


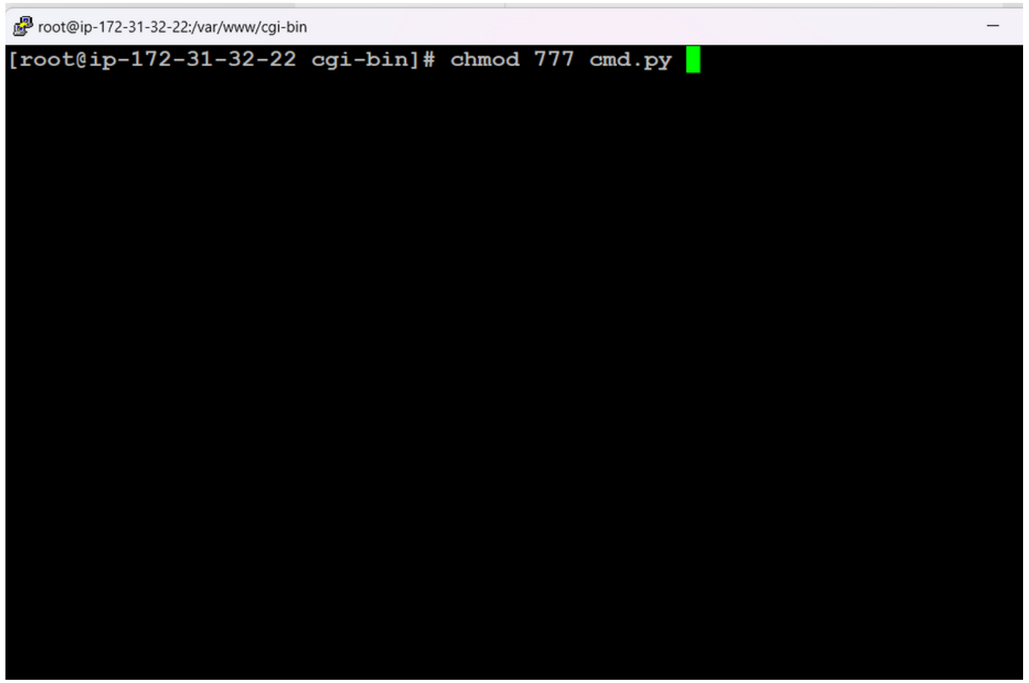
Apache

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***

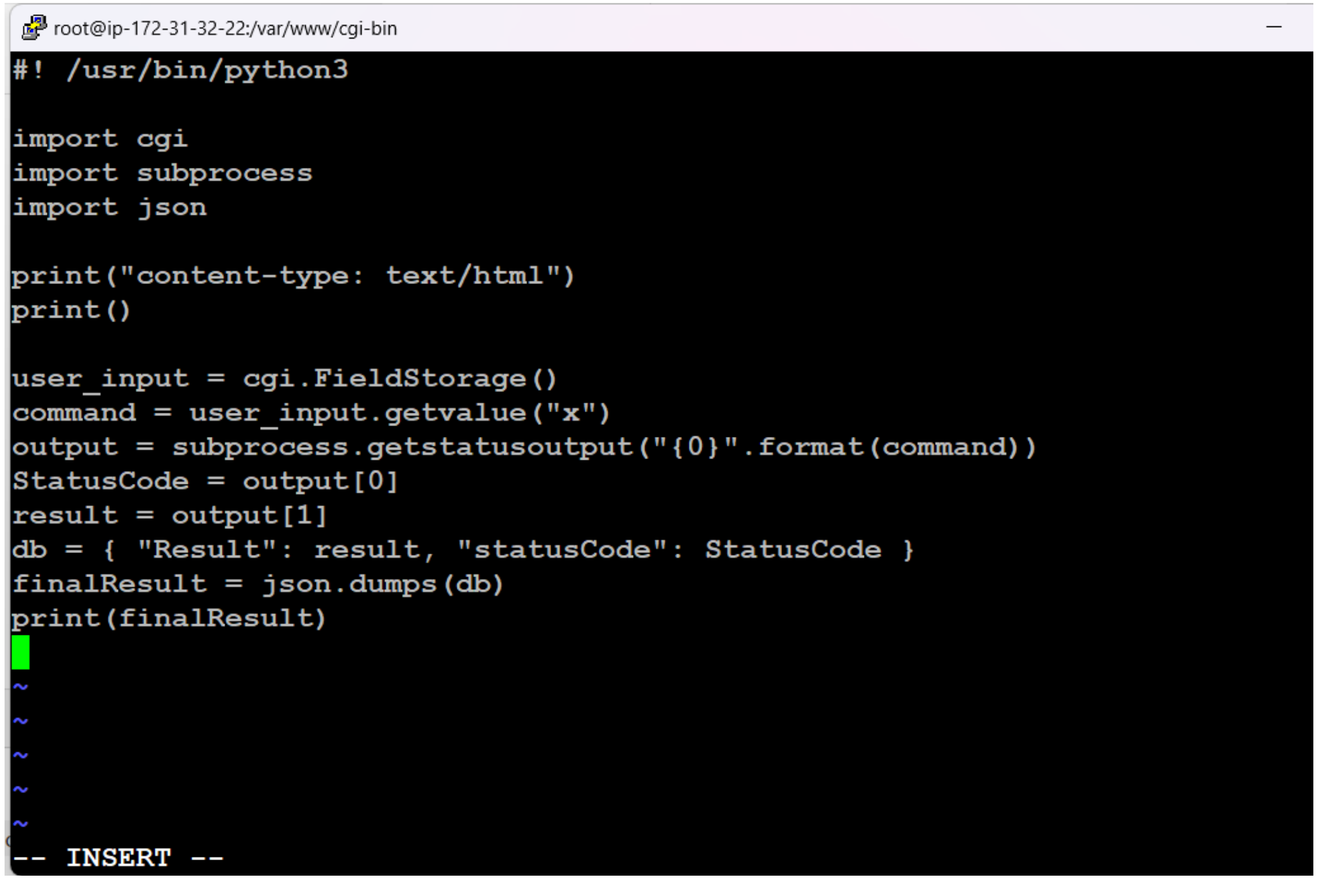
while we also installed the python service as it will be the communicating host in understanding the language of commands and executing the same via webhooks received the command for the same is **yum install python3**





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



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**

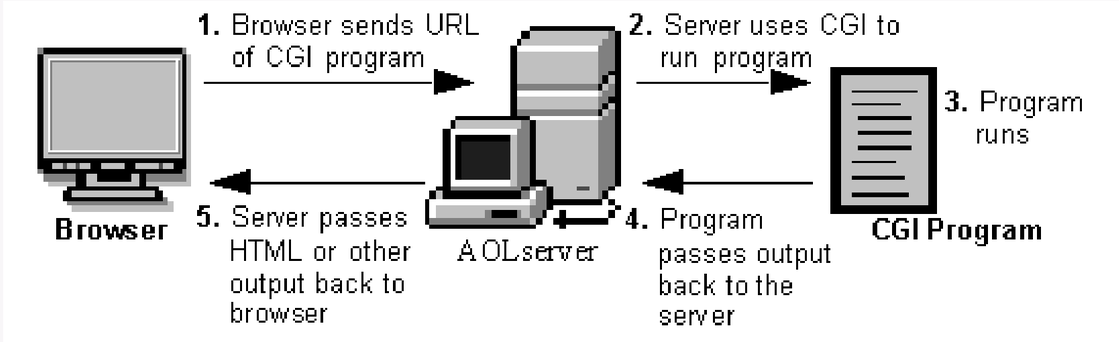
db---------> will create a JSON format and the final result will serialize the same and send back to the host requesting it.

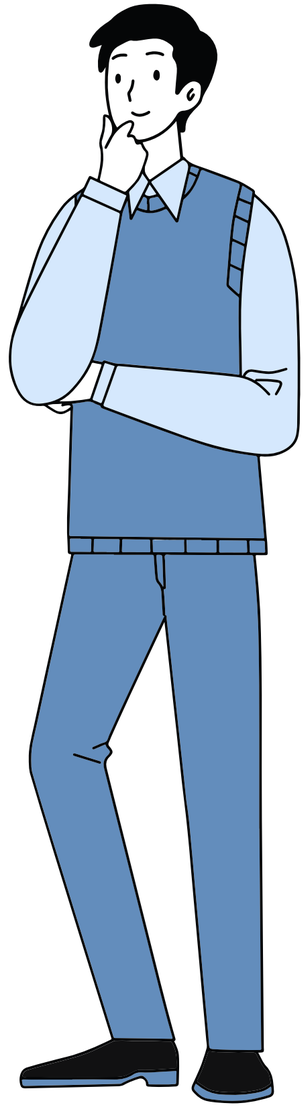
What is CGI(Comman Gateway Interface) and how does it works.....?

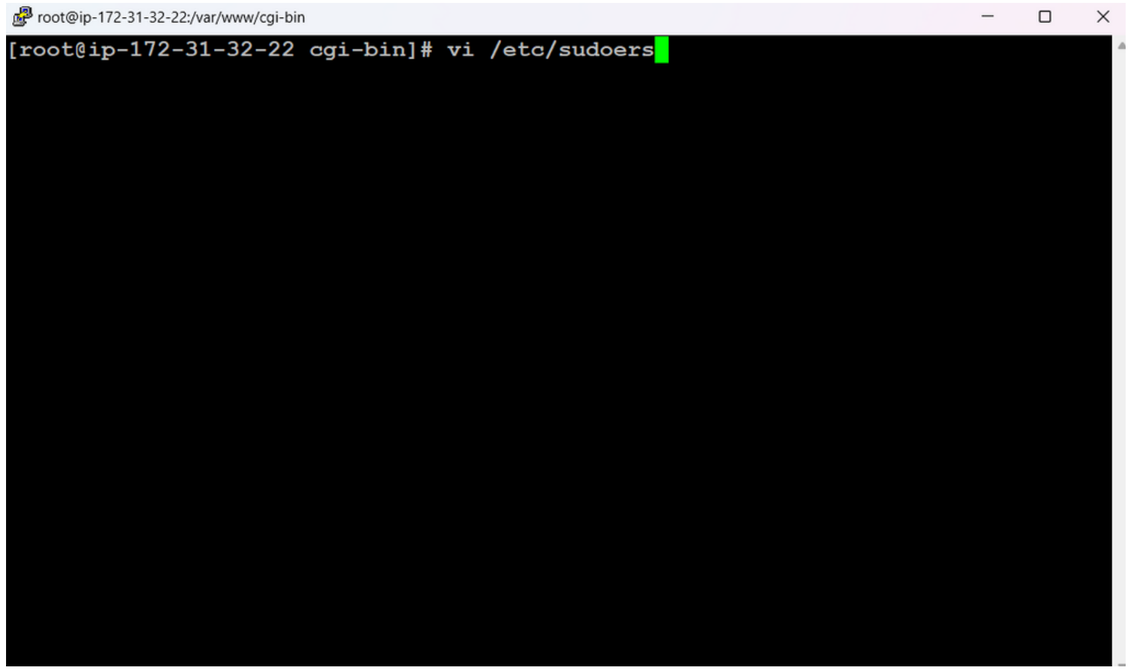
CGI (Common Gateway Interface) is a standard way of running programs from a Web server. Often, CGI programs are used to generate pages dynamically or to perform some other action when someone fills out an HTML form and clicks the submit button.

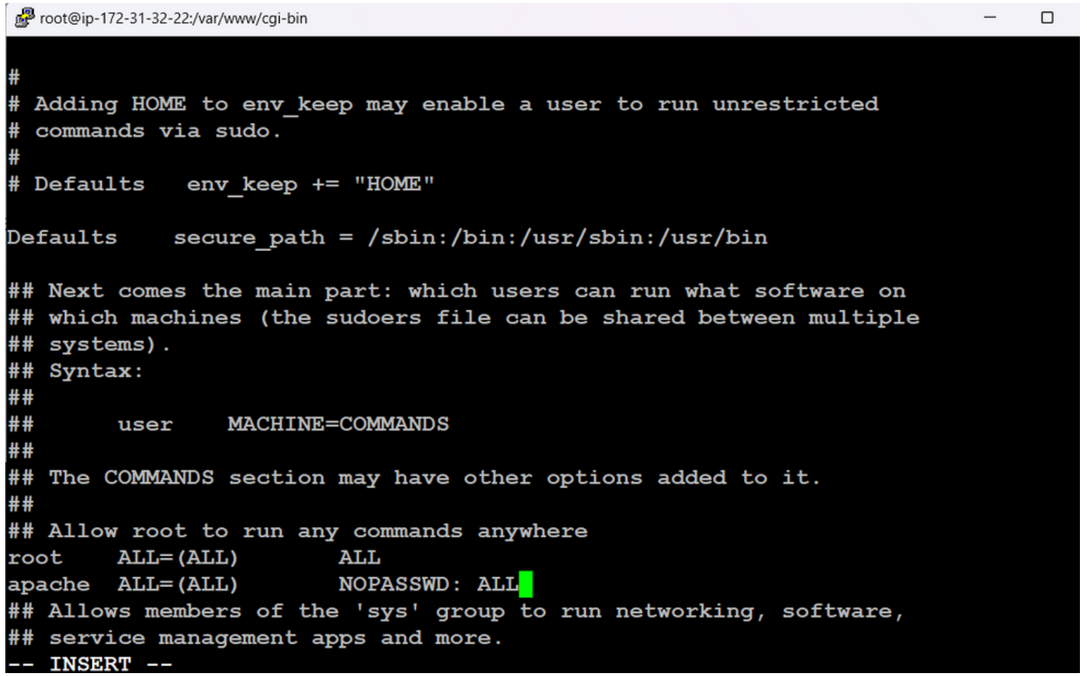
Basically, CGI works like this:

A r**eader 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**.





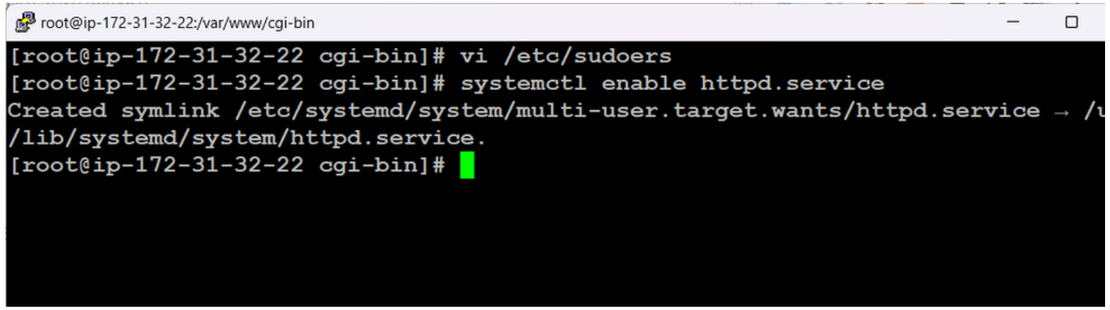


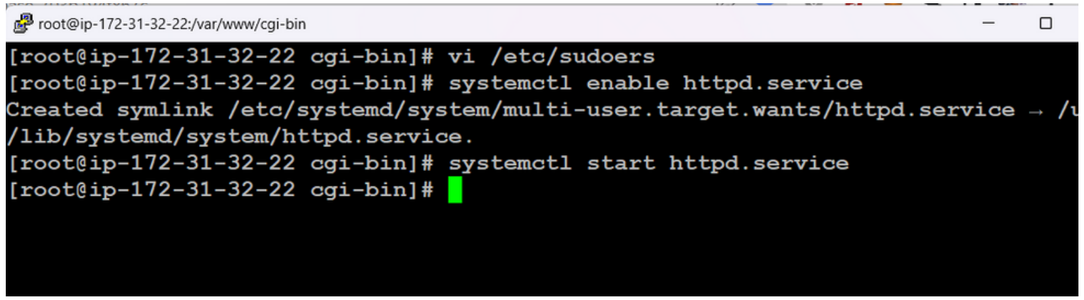


4. Now we need to provide root permissions to the apache server as if we need to provide user-based access to the terminal, not the root user as for security reasons, the command of execution will be **vi /etc/sudoers**

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

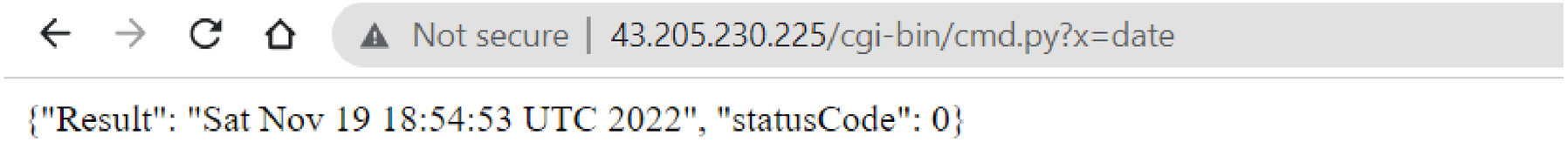
The above line denotes to give all root access to apache while NOPASSWD gives permission to access by any user without any password



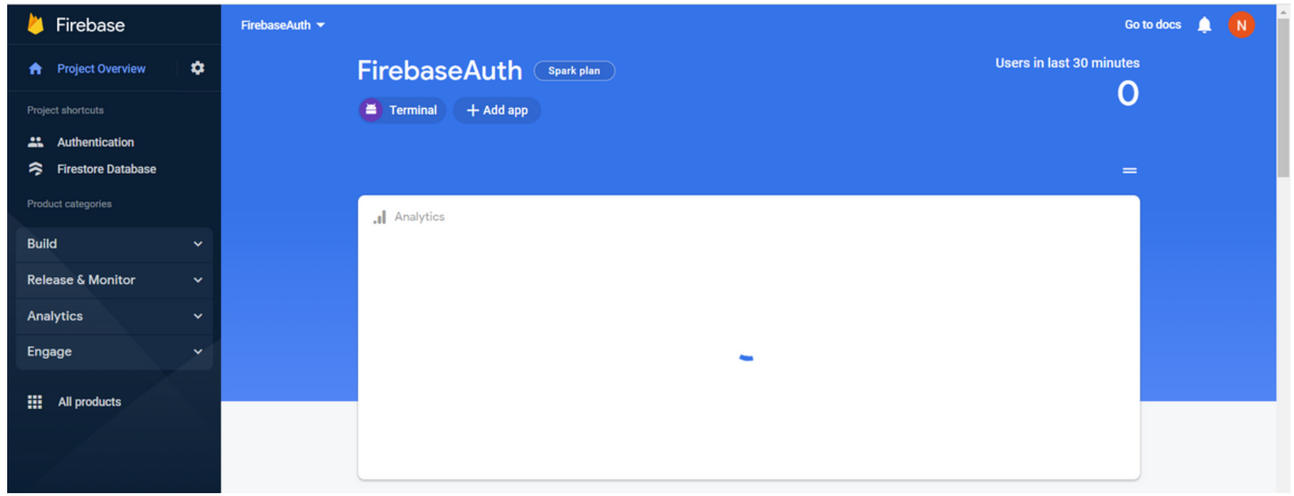


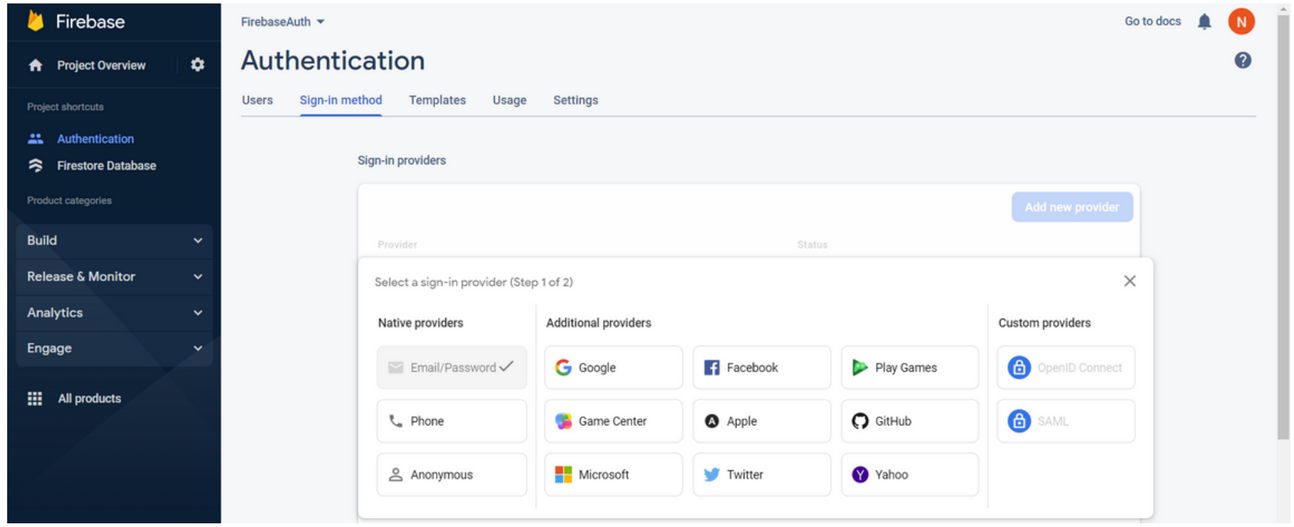
5. Now enable and start the apache server, the command for the same is **systemctl enable httpd.service** and **systemctl start httpd**

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

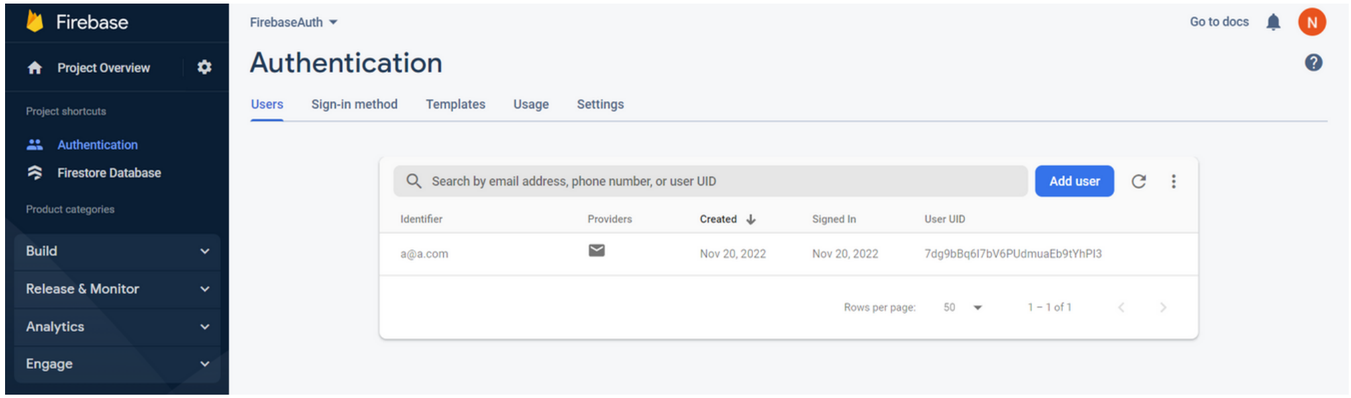


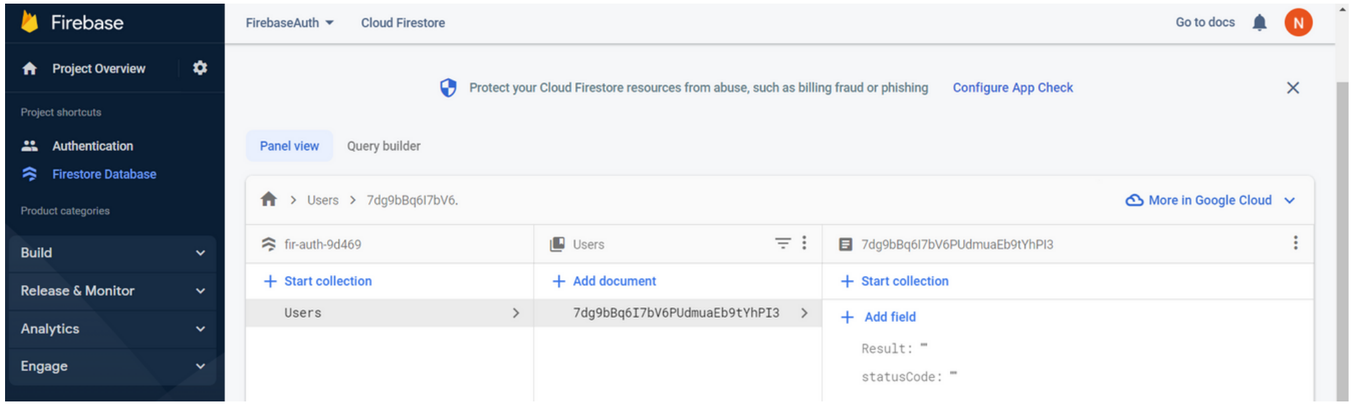


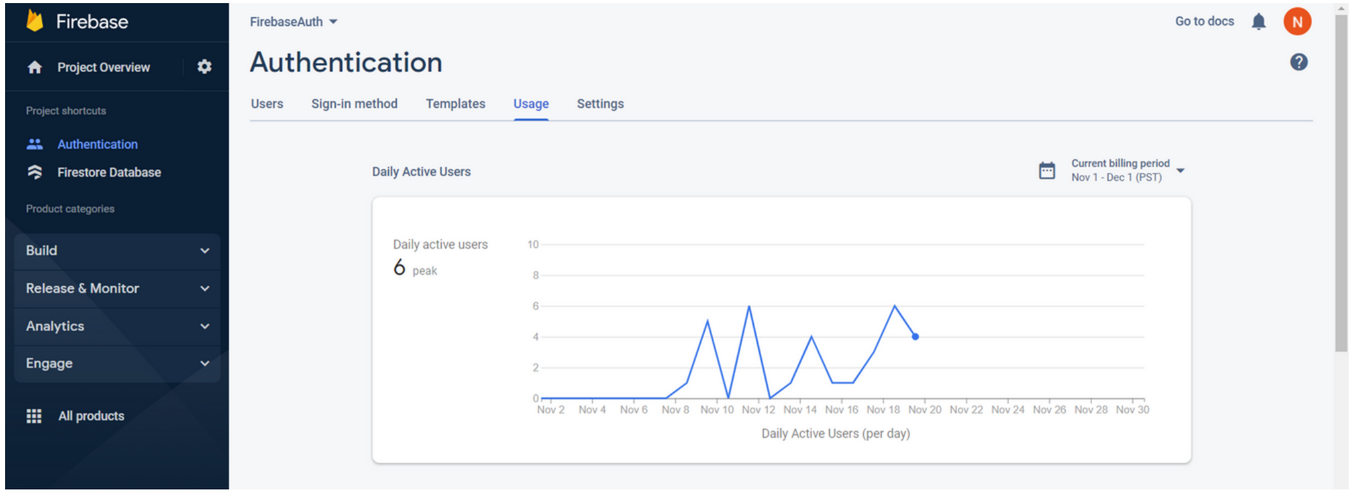
6. Now for the backend in storing the data we need a service that is firebase responsible for handling data requests. **create a firebase project and add authentication and firestore database services to your project.**

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.**







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**.

FRONTEND, BACKEND CONNECTION

3

pubspec.yaml

```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.

Flutter

choice_page.dart

lib > pages > choice_page.dart > choice > build

```dart
Container(
  width: double.maxFinite,
  height: 400,
  child: RiveAnimation.asset(
    "assets/this_that.riv",
    fit: BoxFit.fill,
  ), // RiveAnimation.asset
), // Container
NeoPopTiltedButton(
  isFloating: true,
  onTapUp: () {
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) => LoginPage(),
      ), // MaterialPageRoute
    );
  },
  decoration: const NeoPopTiltedButtonDecoration(
    color: Color.fromARGB(255, 22, 72, 137),
    plunkColor: Color.fromARGB(255, 22, 72, 137),
    shadowColor: Color.fromRGBO(36, 36, 36, 1),
    showShimmer: true,
  ), // NeoPopTiltedButtonDecoration
  child: const Padding(
    padding: EdgeInsets.symmetric(
      horizontal: 70.0,
      vertical: 15,
    ), // EdgeInsets.symmetric
    child: Text(
      'IANT USER',
      style: TextStyle(
        color: Colors.white,
        fontWeight: FontWeight.bold,
        fontSize: 15,
      ), // TextStyle
    ), // Text
  ), // Padding
```

choice_page.dart

lib > pages > choice_page.dart > choice > build

```dart
import 'package:terminal/pages/dashboard/screens/product/terminal_page.dart';
import 'package:terminal/pages/login_register_page.dart';

class choice extends StatelessWidget {
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color(0xff2D5F7B),
      body: Container(
        width: double.infinity,
        height: double.infinity,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            NeoPopTiltedButton(
              isFloating: true,
              onTapUp: () {
                Navigator.of(context).push(
                  MaterialPageRoute(
                    builder: (context) => public_cli(),
                  ), // MaterialPageRoute
                );
              },
              decoration: const NeoPopTiltedButtonDecoration(
                color: Color.fromARGB(255, 22, 72, 137),
                plunkColor: Color.fromARGB(255, 22, 72, 137),
```

auth.dart

lib > auth.dart > Auth > accDelete

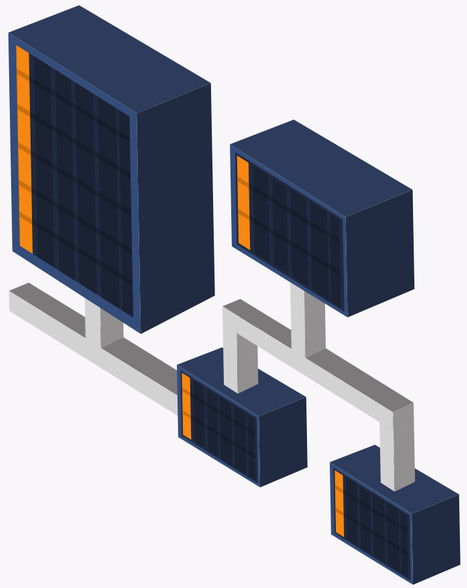
```dart
class Auth {
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;

  User get currentUser => _firebaseAuth.currentUser;

  Stream<User> get authStateChanges => _firebaseAuth.authStateChanges();

  Future<void> signInWithEmailAndPassword({
    String email,
    String password,
  }) async {
    await _firebaseAuth.signInWithEmailAndPassword(
      email: email,
      password: password,
    );
  }

  Future<void> createUserWithEmailAndPassword({
    String email,
    String password,
  }) async {
    await _firebaseAuth.createUserWithEmailAndPassword(
        email: email, password: password);
  }

  Future<void> signOut() async {
    await _firebaseAuth.signOut();
  }

  Future<void> accDelete() async {
    FirebaseFirestore.instance
        .collection("Users")
        .doc(FirebaseAuth.instance.currentUser.uid)
        .delete();
    await _firebaseAuth.currentUser.delete();
  }
```

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

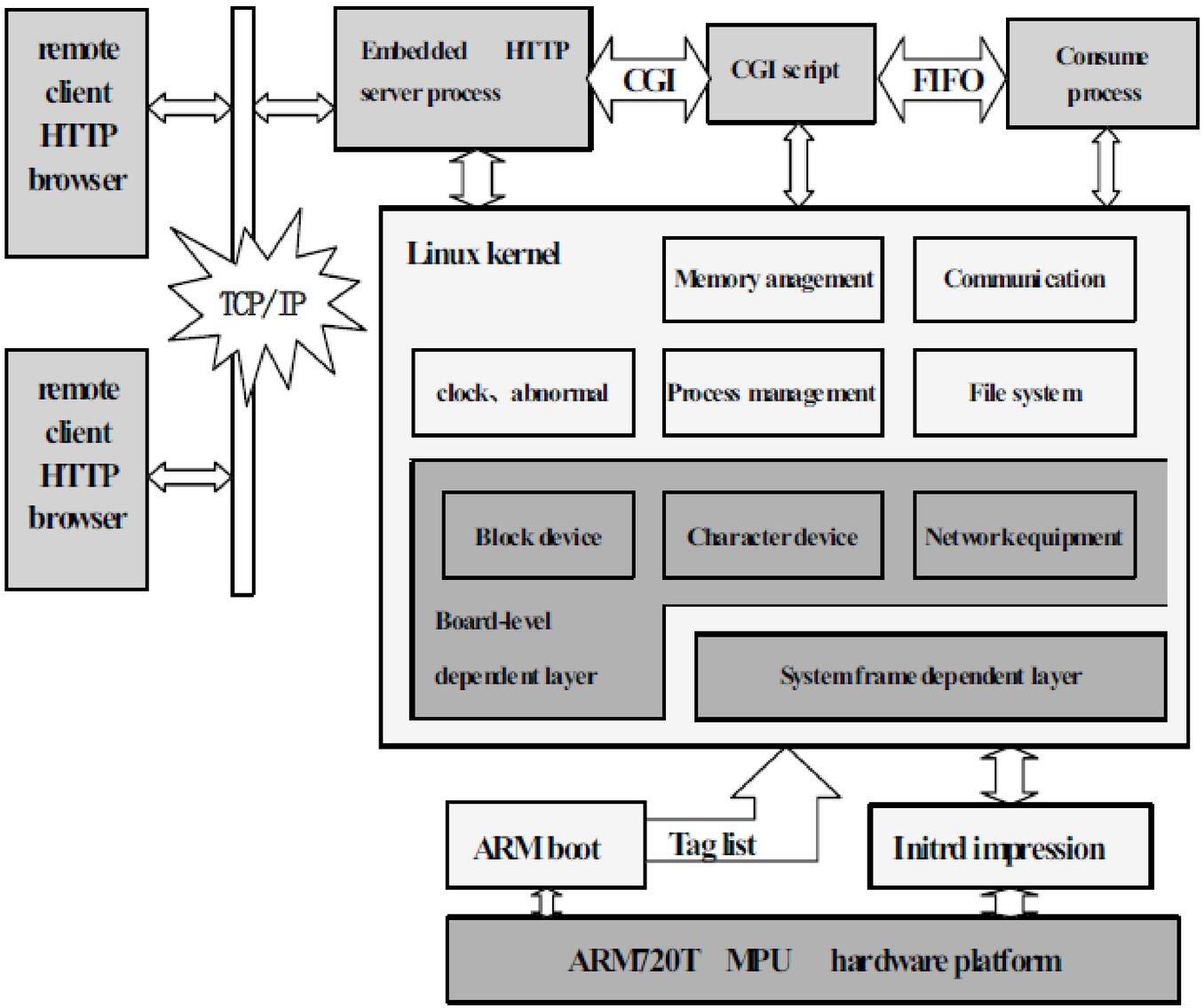
terminal_page.dart

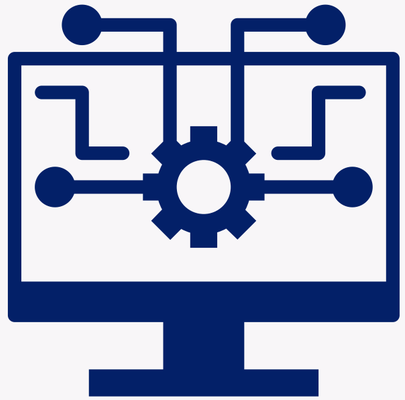
lib > pages > dashboard > screens > product > terminal_page.dart > _BodyState > getOutput

```dart
class _BodyState extends State<Body> {
  String terminalopt = 'IN_Terminal';
  final _formKey = GlobalKey<FormState>();
  final firestoreInstance = FirebaseFirestore.instance;
  var firebaseUser = FirebaseAuth.instance.currentUser;
  TextEditingController command = TextEditingController();
  var input;
  var url = '';
  @override
  void initState() {
    super.initState();
    firestoreInstance.collection("Users").doc(firebaseUser.uid).set({
      "Result": "",
      "statusCode": "",
    });
  }

  getOutput(input) async {
    if (terminalopt == "IN_Terminal") {
      url = 'http://43.205.195.2/cgi-bin/cmd.py?x=$input';
    } else if (terminalopt == "US_Terminal") {
      url = 'http://3.87.141.161/cgi-bin/cmd.py?x=$input';
    }
    var result = await http.get(Uri.parse(url));
    var output = json.decode(result.body);

    await firestoreInstance.collection("Users").doc(firebaseUser.uid).set({
      "Result": output['Result'],
      "statusCode": output['statusCode']
    }).then((_) => print("Sucess"));
  }
```



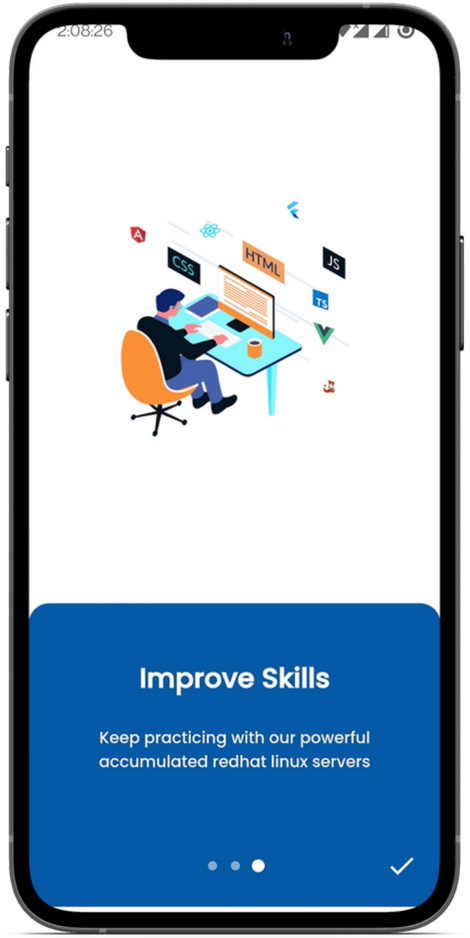
3. In this code we have initiated the command request over **HTTP**, which will be **triggered** to the **IP** we have provided i.e. of our **Linux machines** but with conditions for the **available zones** where we created our machines for user convenience and **load balancing** over the same



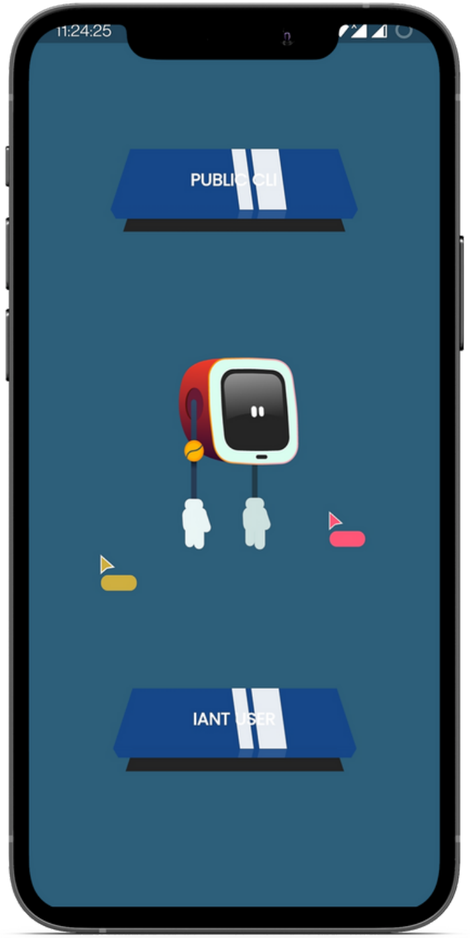


Server architecture and its **internal working** how our **python script** is being converted and processed to the **Linux shell** and upon execution **revert the response** of the same over **TCP/IP protocol** by reliability check on the same

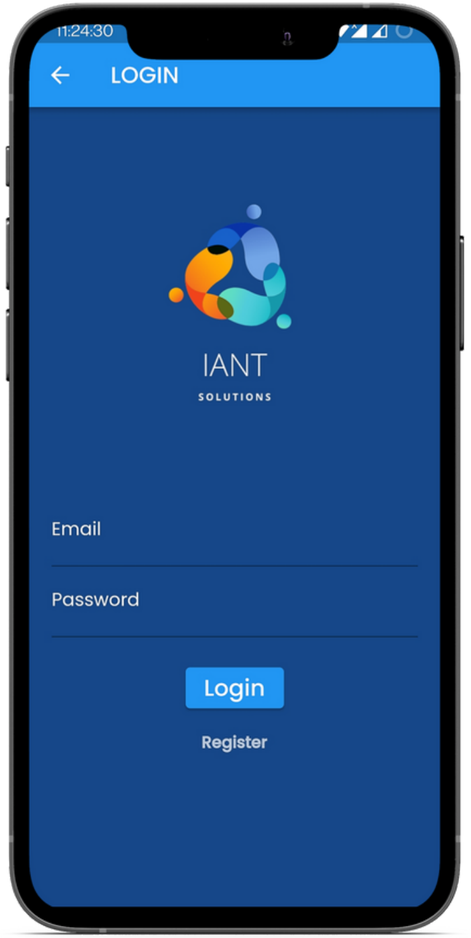
APPLICATION



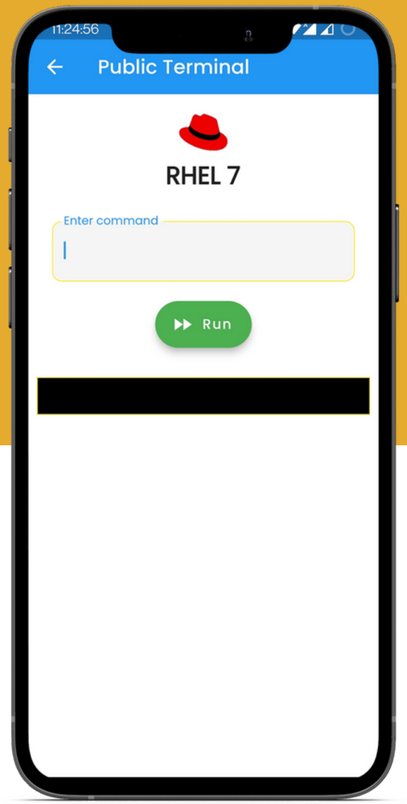
On boarding carousels for first time launch



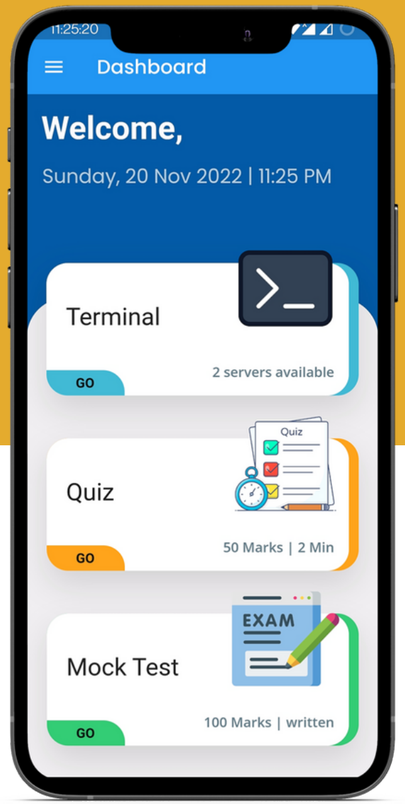
Choice screen selection done by user



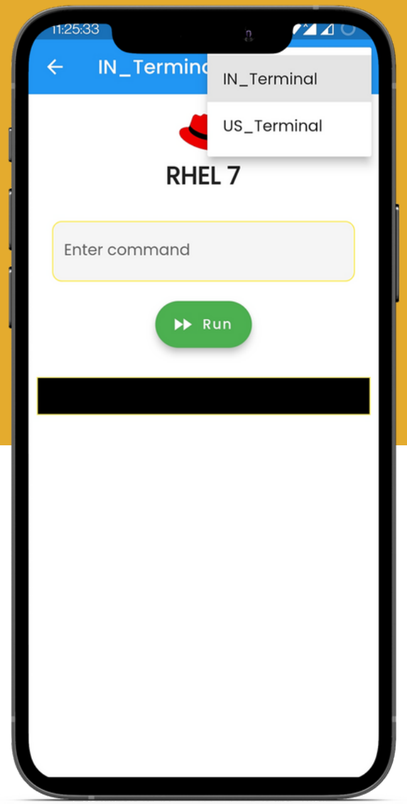
Login screen and register on the same screen



Public terminal used for anytime execution with no restrictions

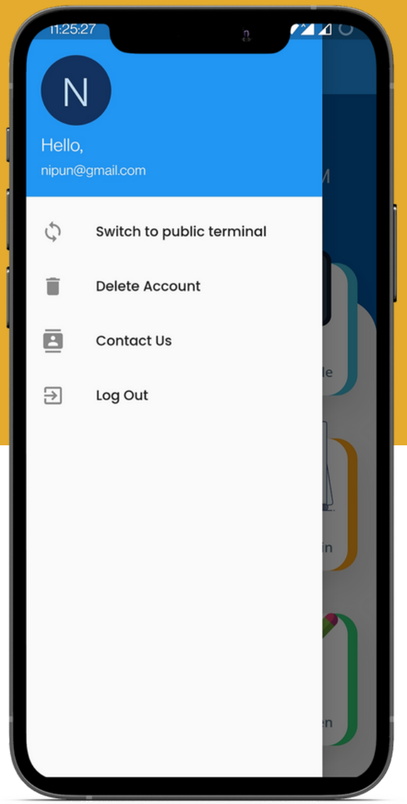


IANT user dashboard on successful login providing quiz and test options

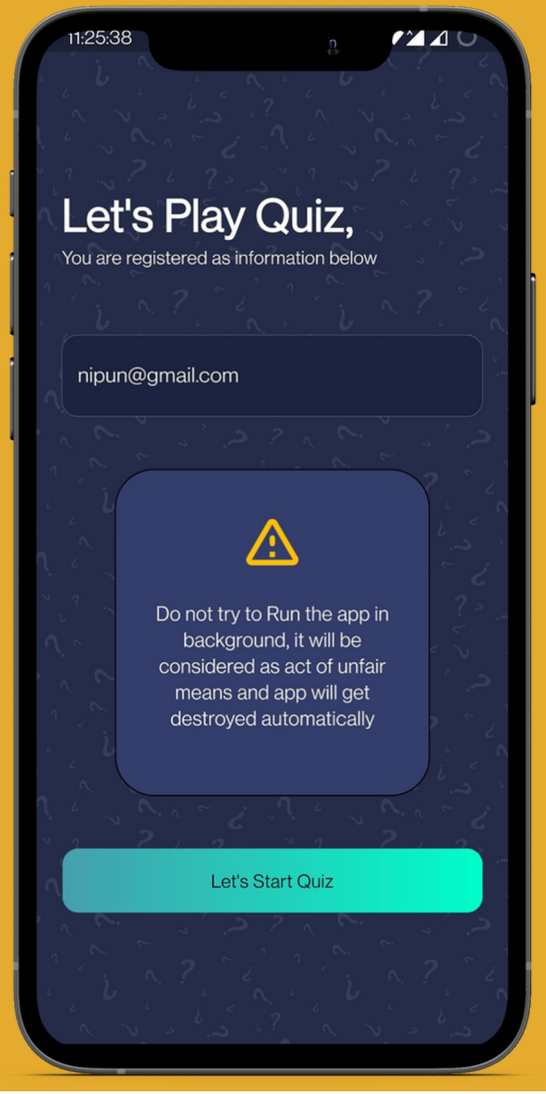


IANT terminal used by users anytime, available in different zones with certain restrictions

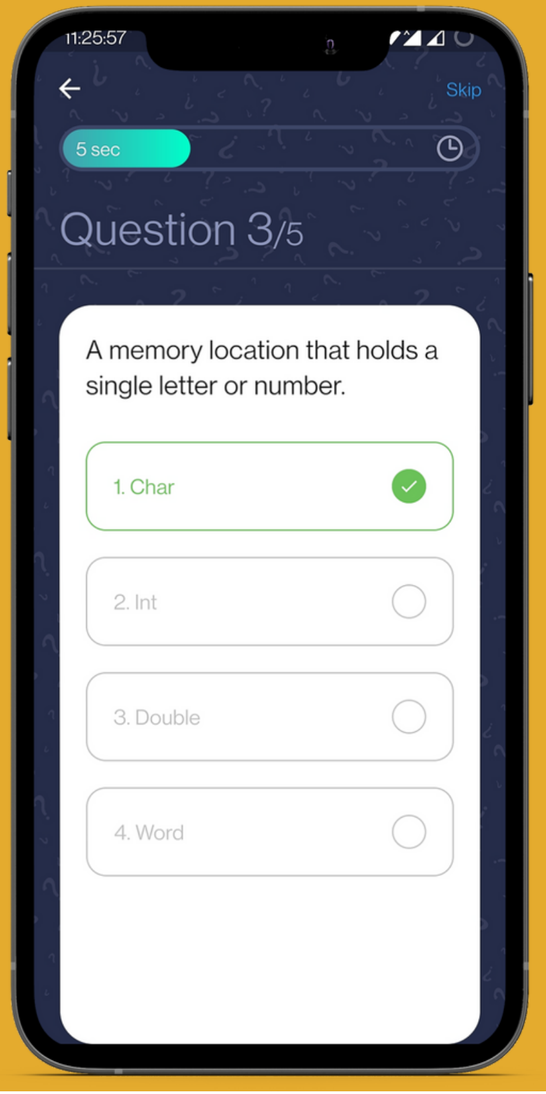
Contact page for anytime accessibility and different links



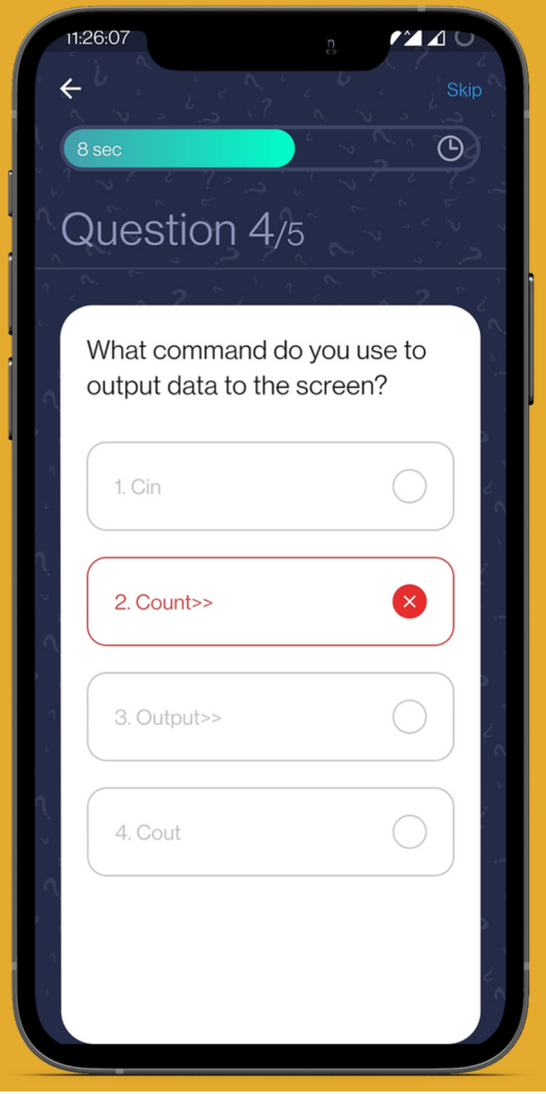
Control page for logout, delete and other features



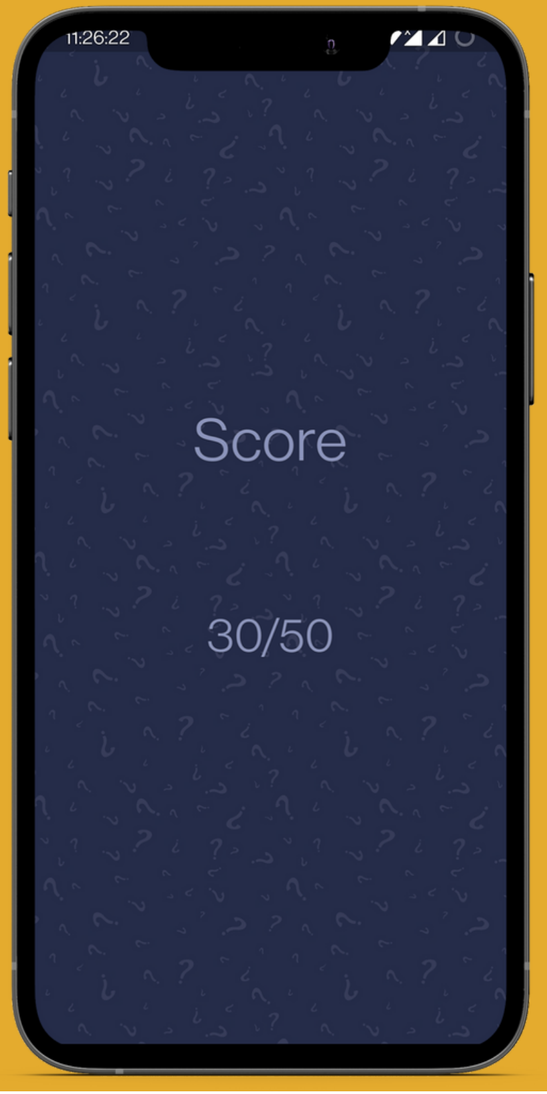
Quiz screen with warning



correct answer selection in time

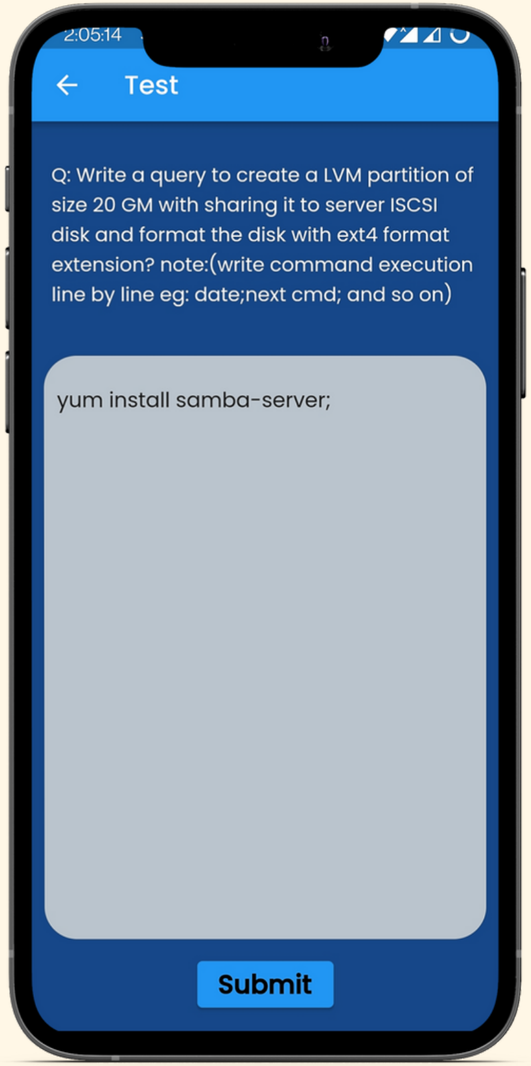


Incorrect answer with no points

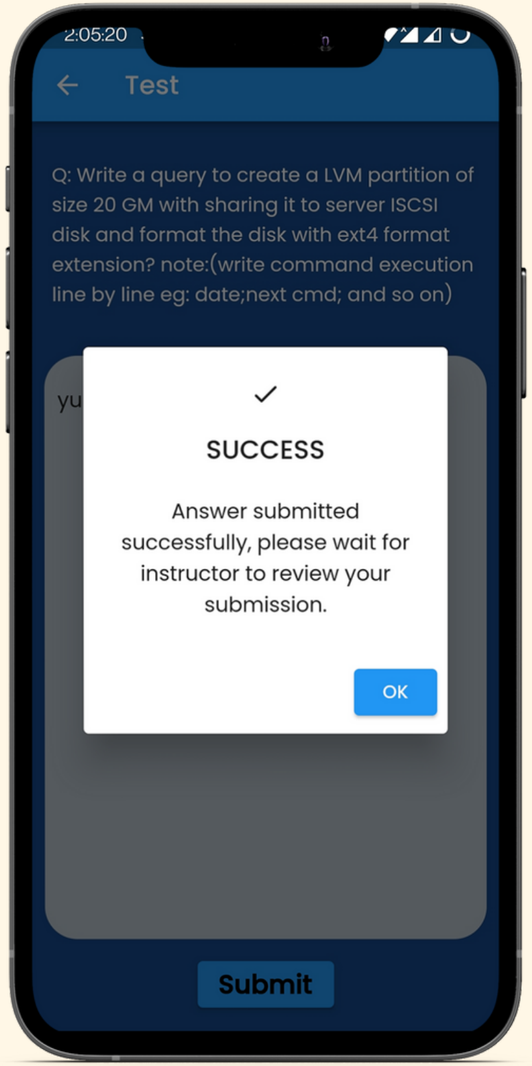


Quiz Score at successful completion

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 reponsiveness 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

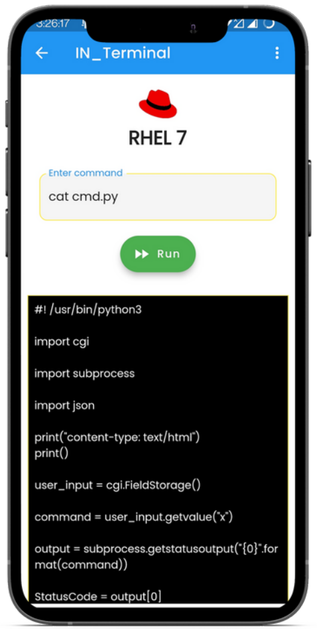
Yes the apps getting crashed and getting locked unless the correct password is granted only when you launch the quiz and try to go to background in the mobile phone

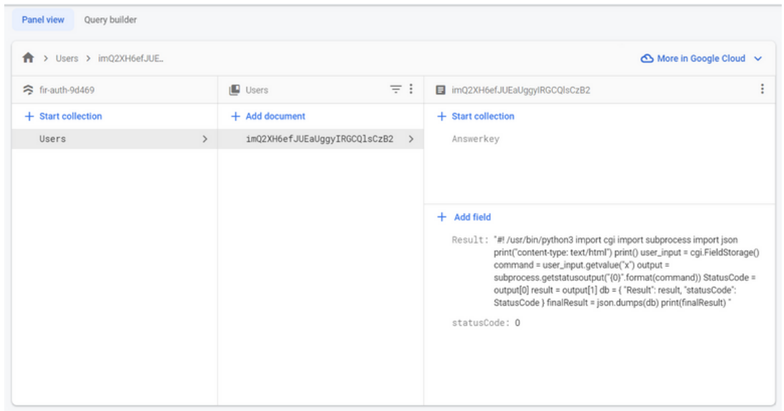
Passed

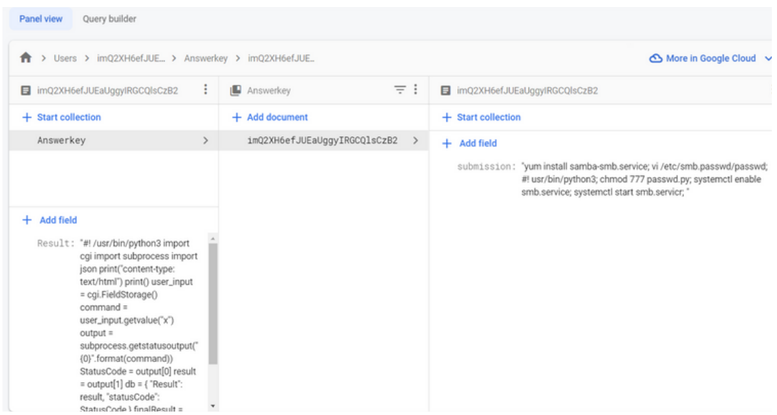
Case 3: User commands and test answers storing in database

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

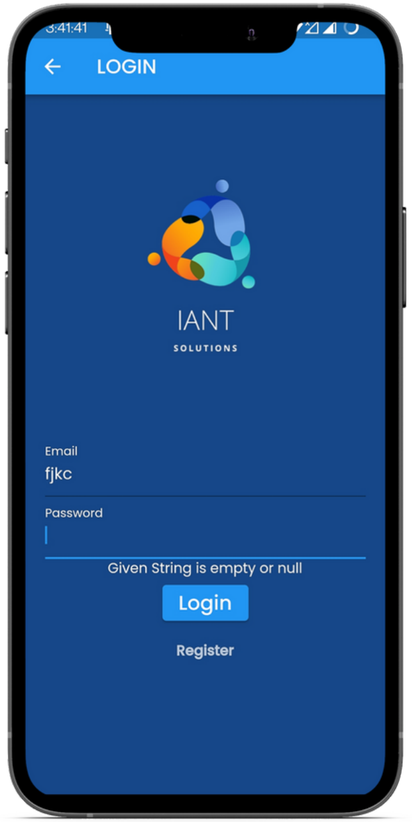


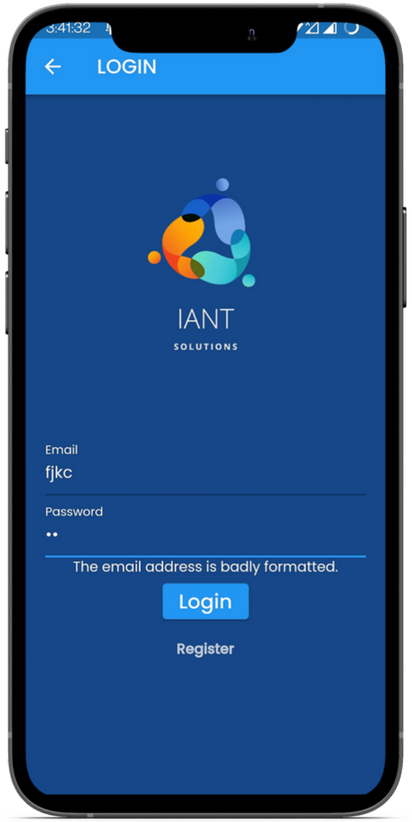


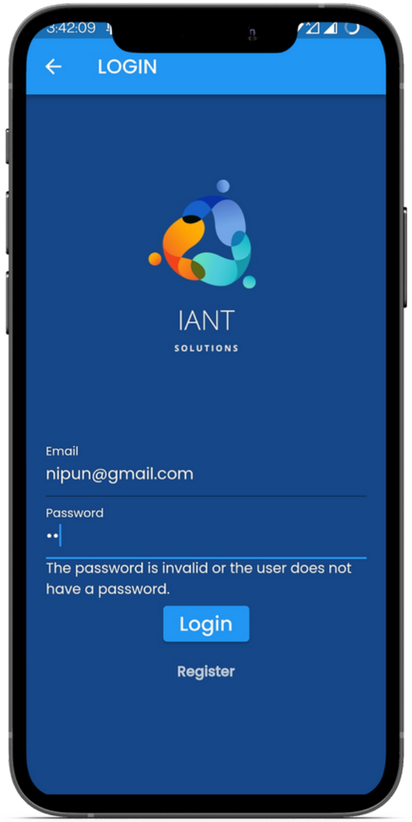


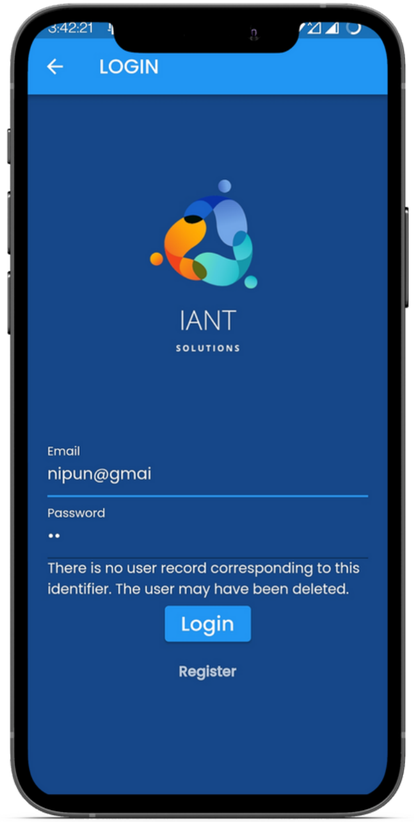
the reflection of data is seen in the database of firebase service as well as seen in the mobile application

Case 4: Login validations working perfectly





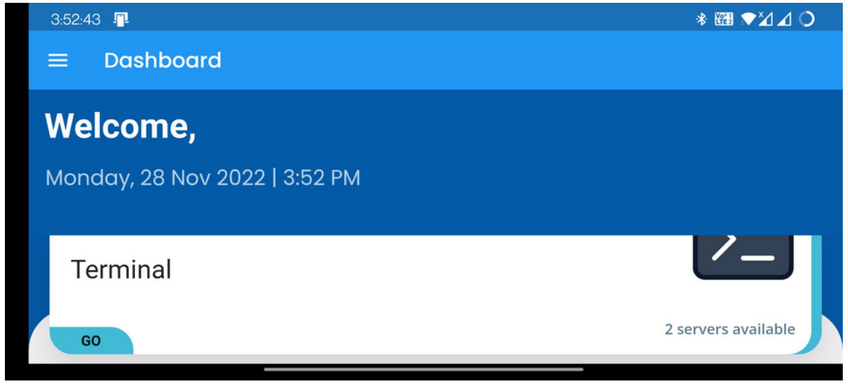


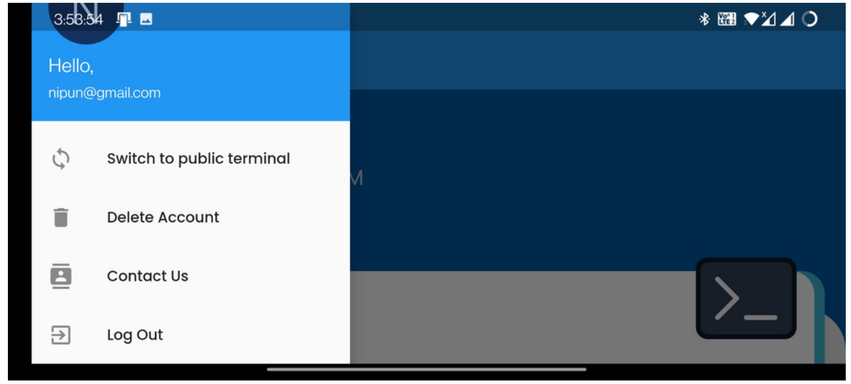


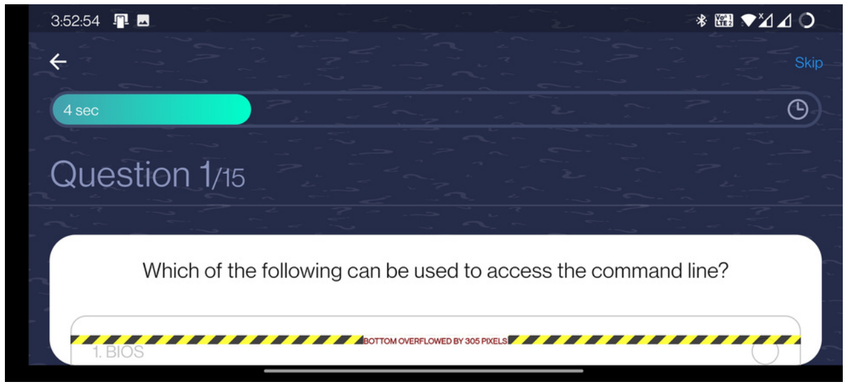
All the validations are working perfectly, from providing the empty strings to providing wrong passwords as well as non-existing user details in the credentials is handled by inbuilt firebase validation feature

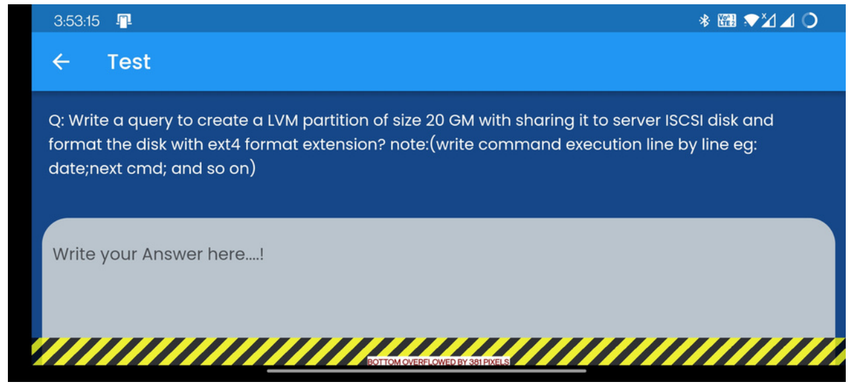
Passed

Case 5: App reponsiveness 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.

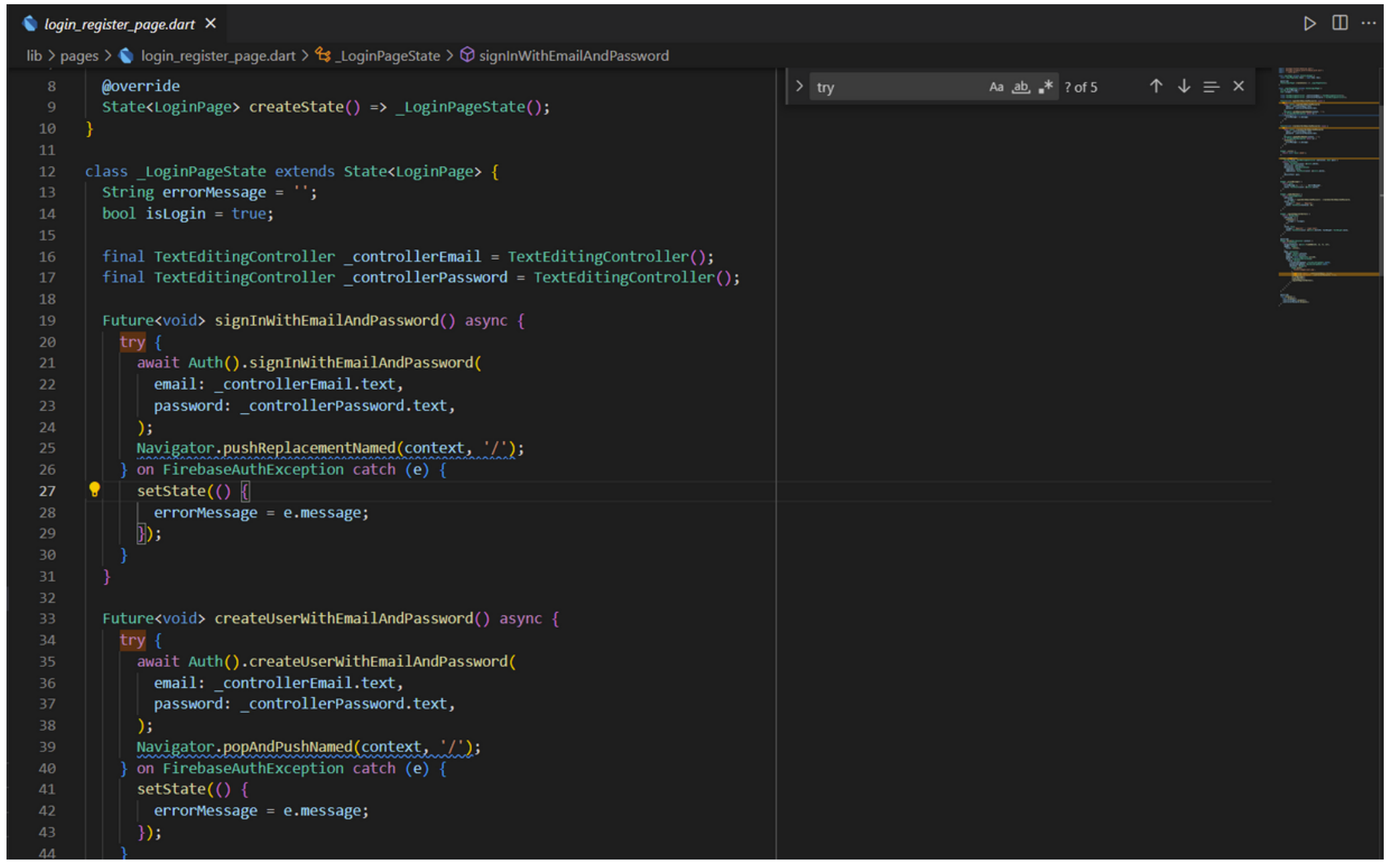
Failed ✕

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.



Passed

CLIENT TESTING

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

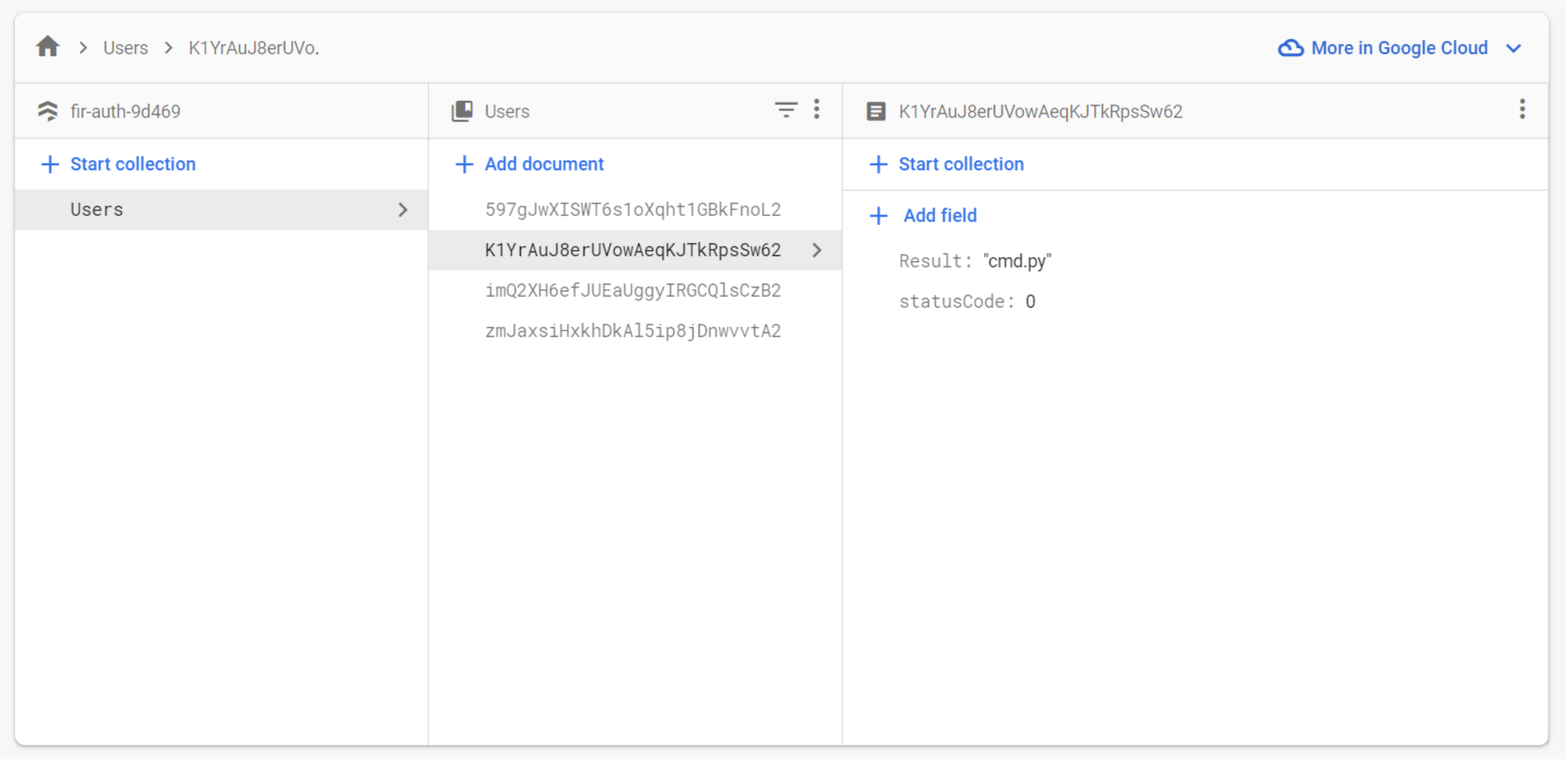
We have tested with the client and performed many operations from basic to complex from different zones as well as with the same zone so the output is coming out to perfection in many cases complex queries need resolution if possible but as AWS constraint rectification require charges to configure.

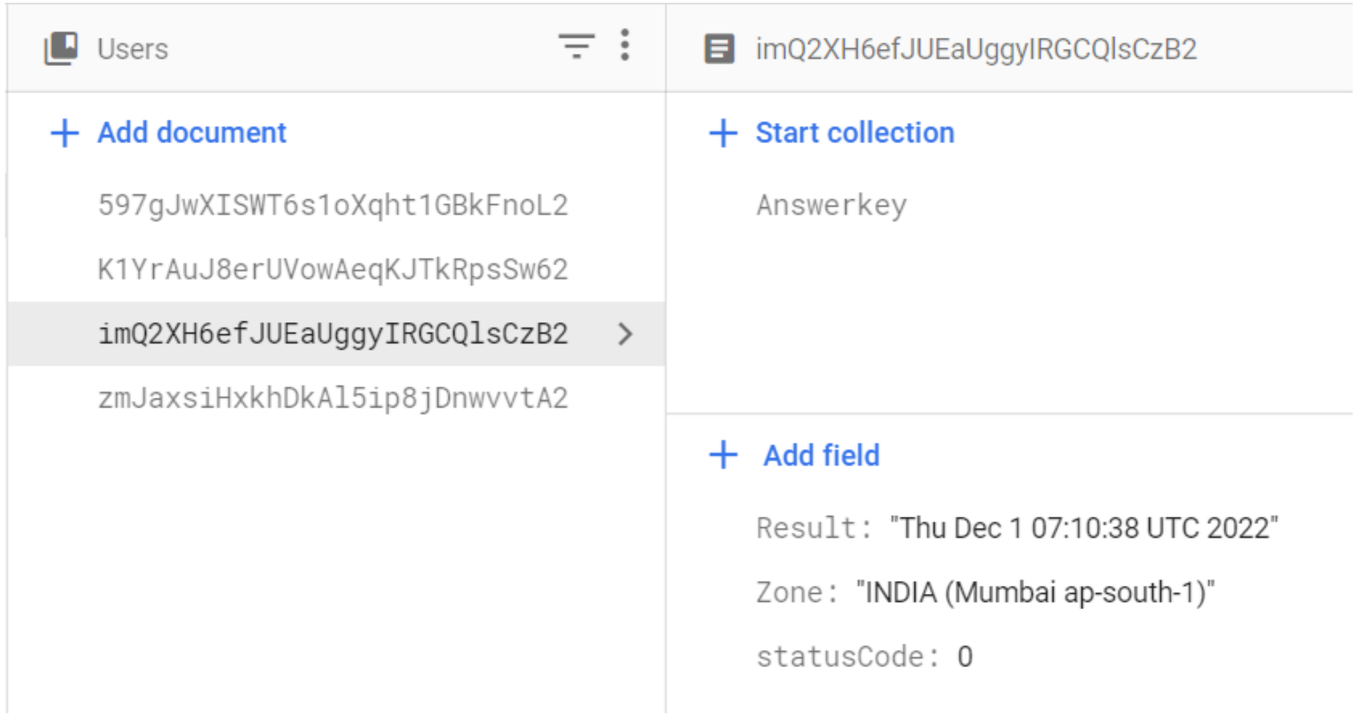
STATUS

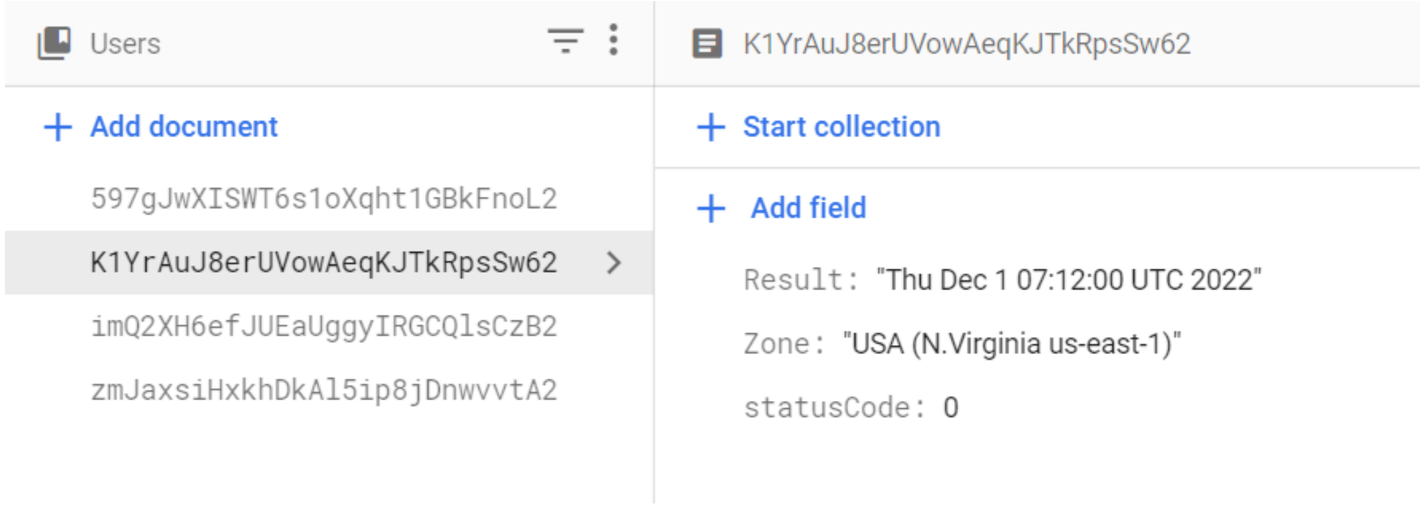
Moderate

Case: **Checking the load balancing/ machine performance**

UNCLEAR

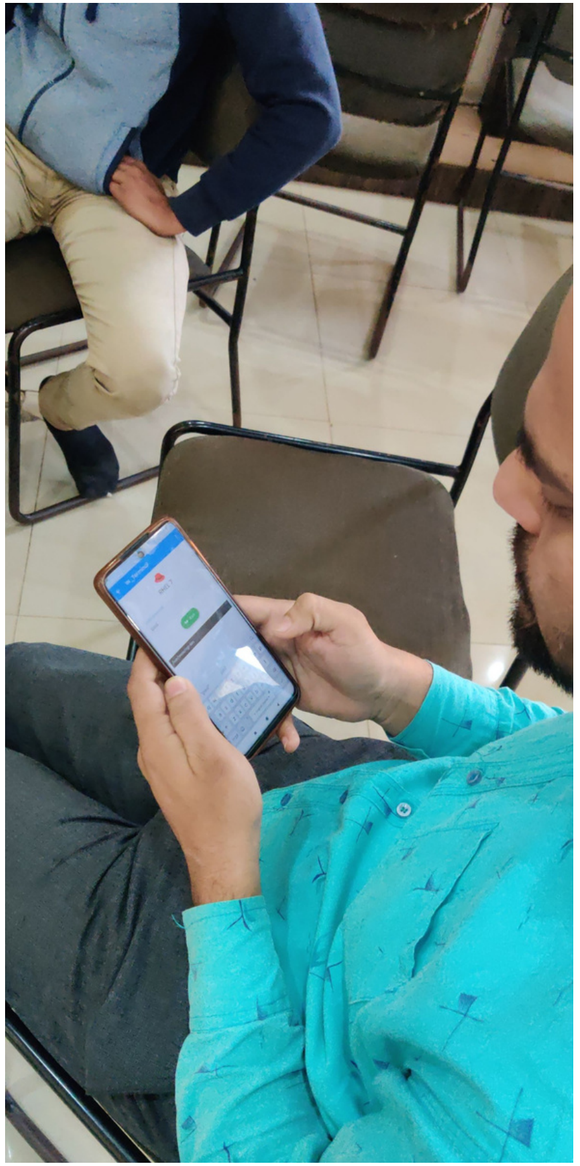






For the load balancing part, we doesn't have a sufficient no. of users to test upon but for the machine performance we checked with a few users, and doesn't seem to have any issues in speed or performance

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

```
  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-ldb-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

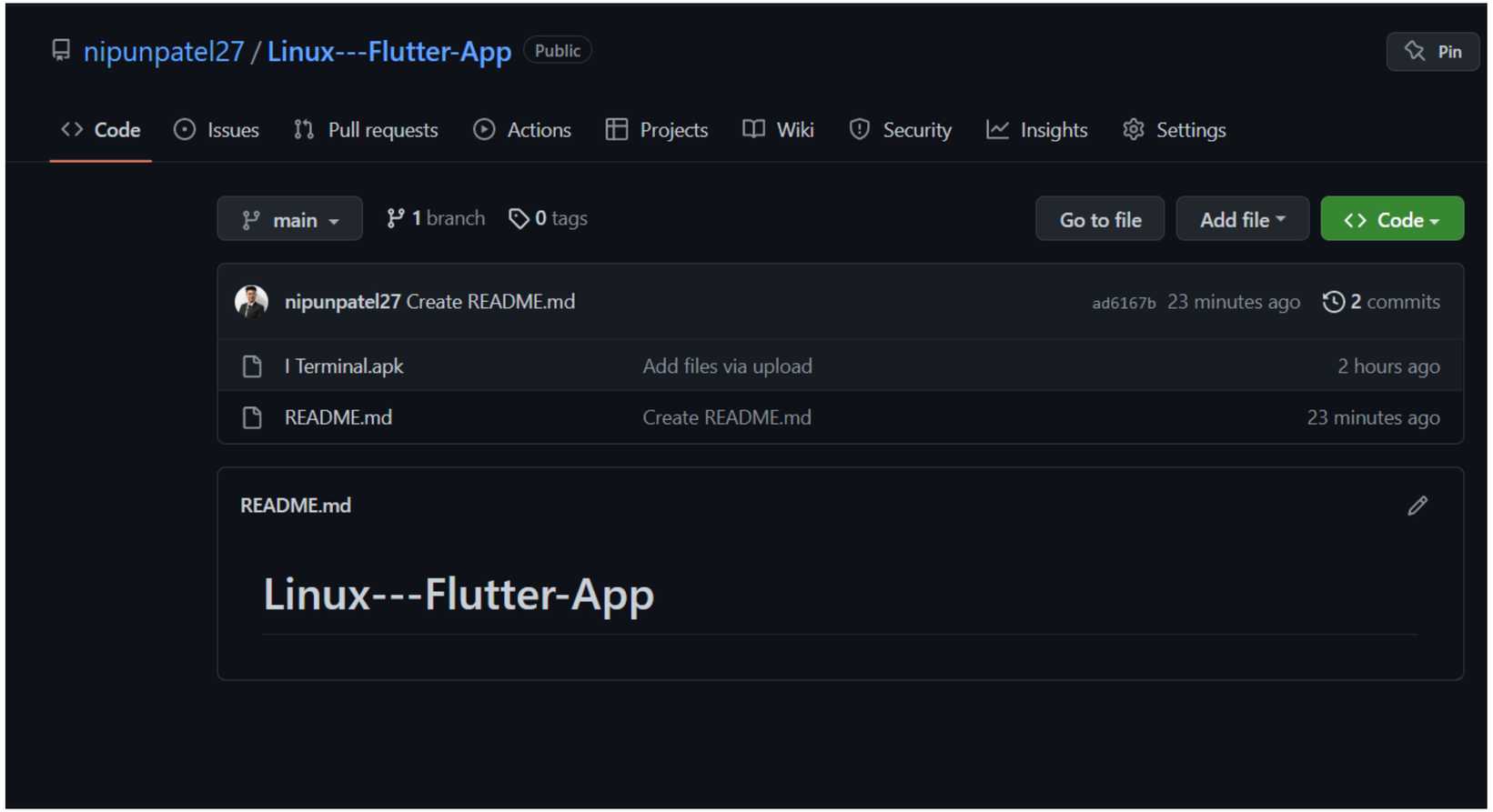
all the Iant user's data is being monitored and saved in the database as per the client-given requirement and ACL permissions are working perfectly for the user to protect the server security, all the exceptions are also handled for login or bad gateway requests passed such as null values in a command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

01	CGI CONSTRAINT	We cannot perform high end command configurations as CGI doesn't support it
02	FIREBASE MONITOR	Firebase provides a monitor for limited no. of logins, beyond that it is a chargeable facility
03	AWS LIMITATION	cannot run multiple EC2 instance as free tier offers only 750 free hours running
04	LOAD BALANCING	It gets difficult to identify load zone when multiple user work on same EC2 zone
05	SCALABILITY	Scaling this application will cost server maintenance charges

DEPLOYMENT



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.