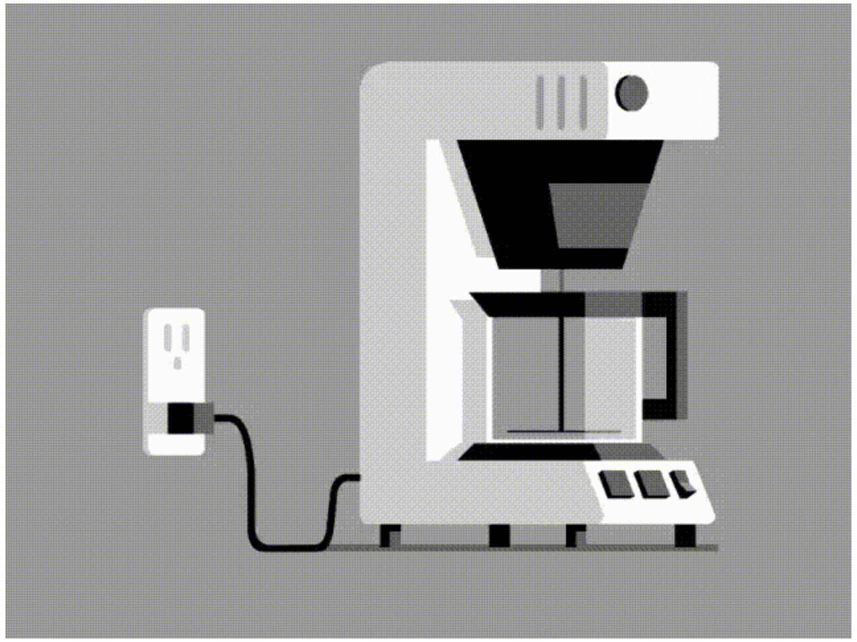
Smart Pour

GROUP 16

E/17/122 - Shazna
E/17/153 - Odasara
E/17/294 - Mishel



Overview

Problem & Solution

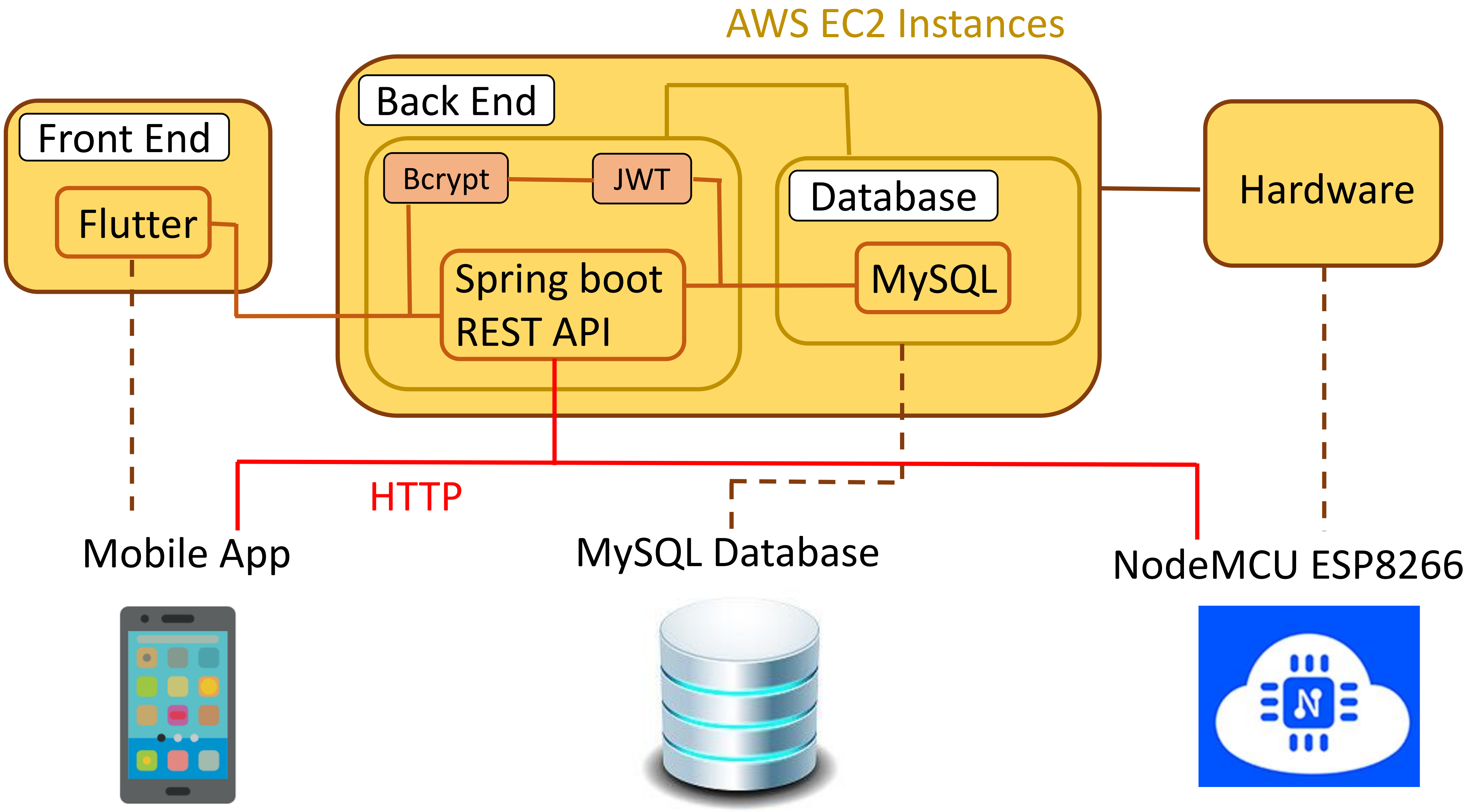
Busy Schedules

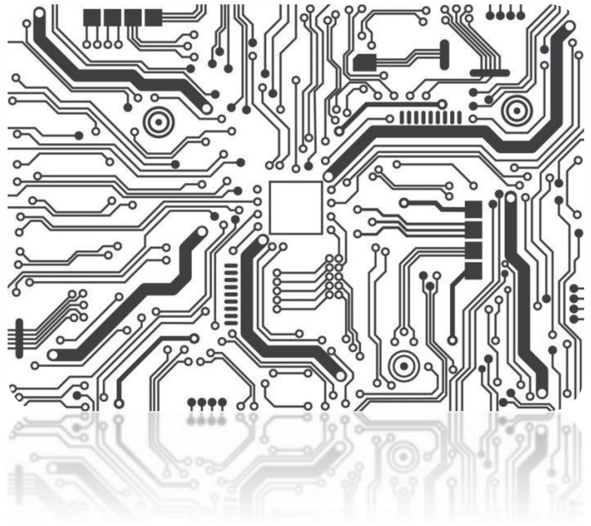
Long queues in cafeterias

Inability to get coffee according to the preference

An automated coffee machine that can be controlled through a mobile application – **"Smart Pour"**

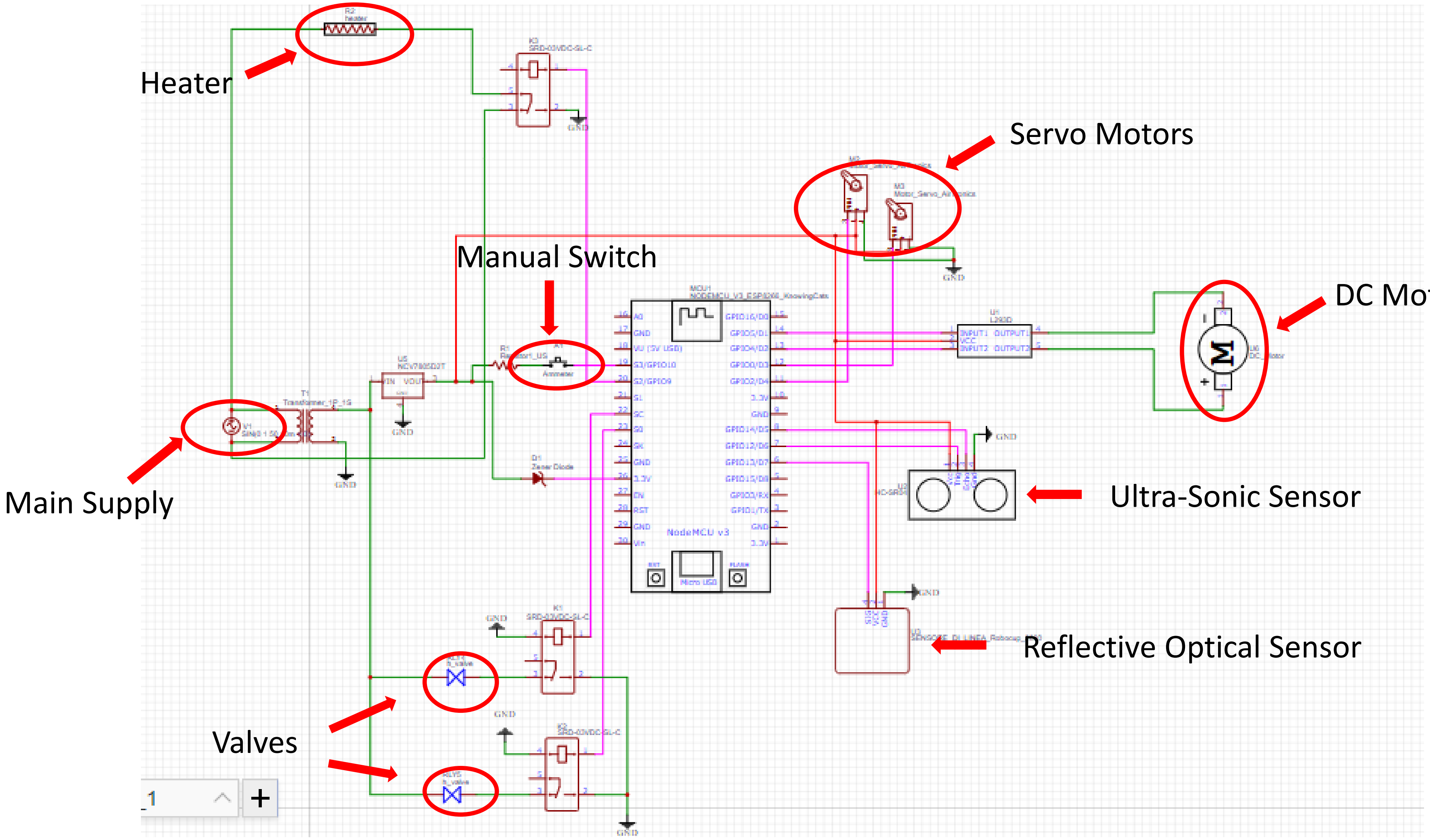
Solution Overview



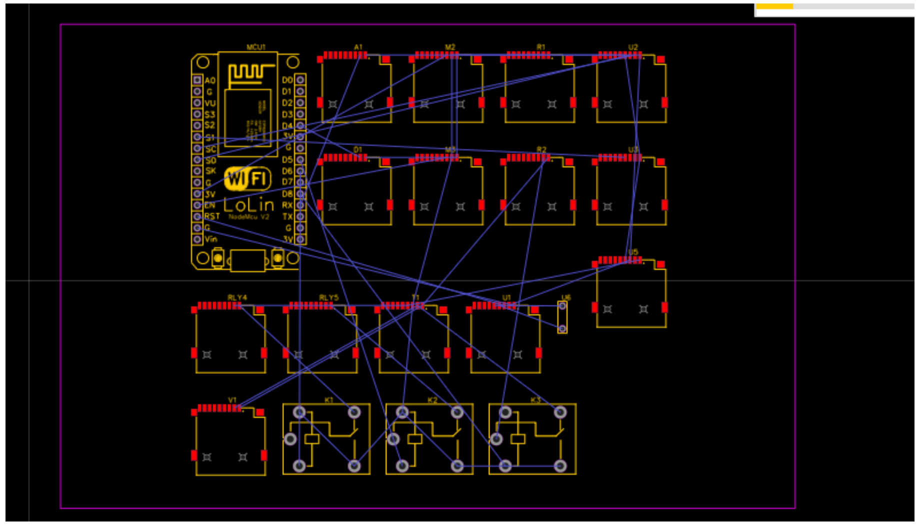


Hardware

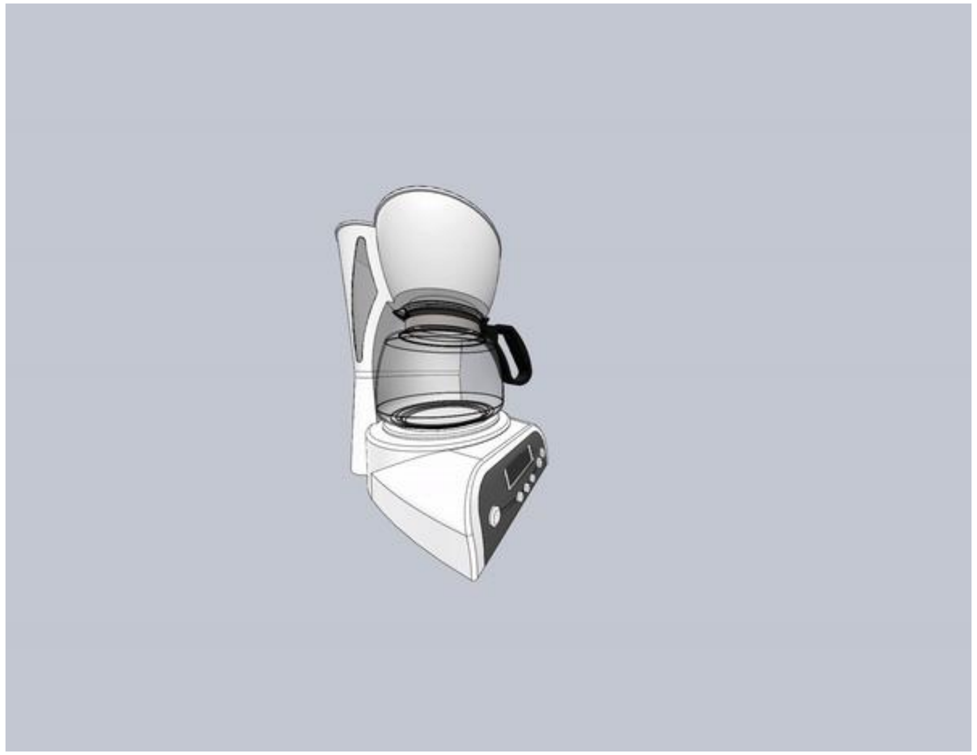
Hardware Designs



Hardware Designs



PCB Layout



3D Model

Functionalities

Security

- Password Authentication to access the storage unit.
- Using insulators to prevent overheating components d ue to the boiling unit.

Reliability

- Manual mode to operate when there is a network failure.

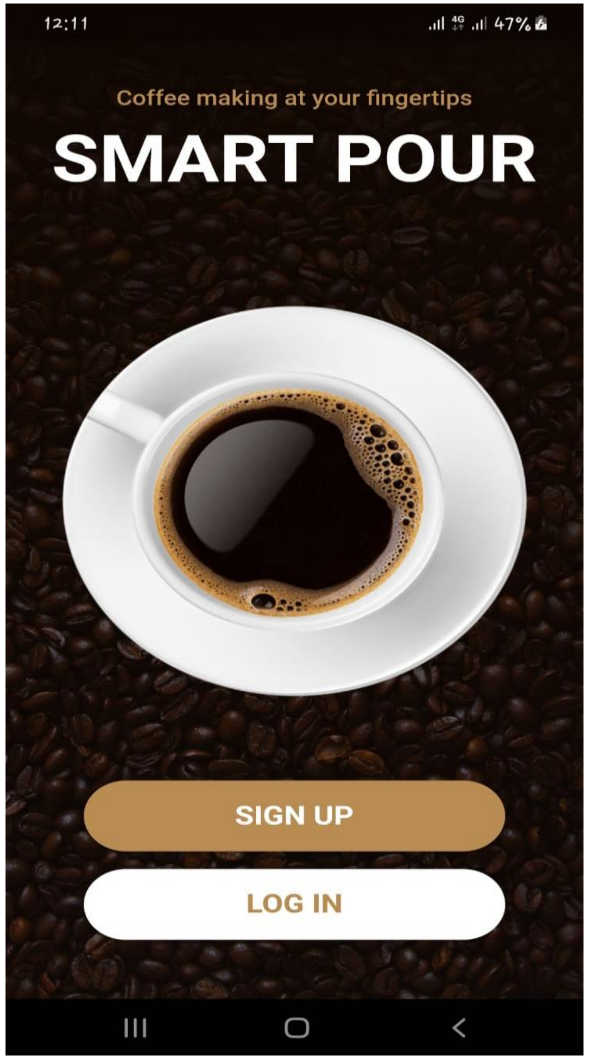
Scalability

•The same design can be repeated to make different drinks.

UI UX

Front end & Back end

Front end

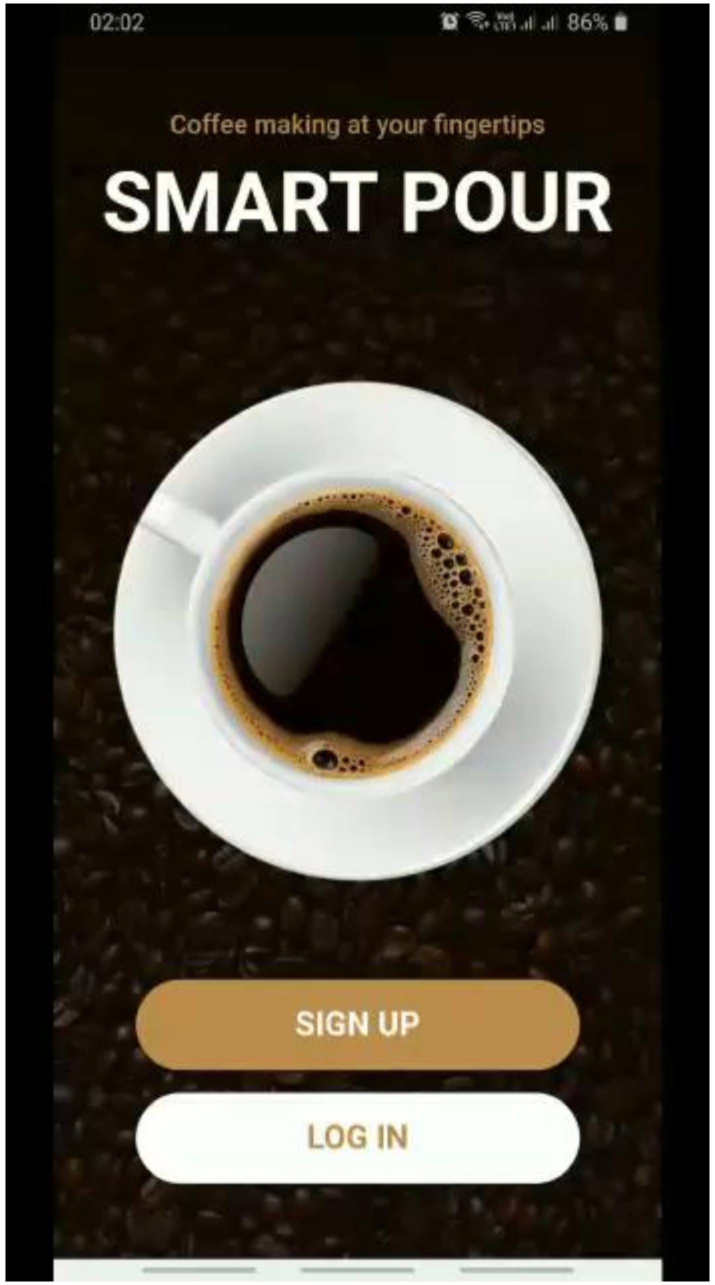


- Get logged
- Add new devices
- Make reservations
- Track ingredients
- Check schedules
- Save favourite recipes

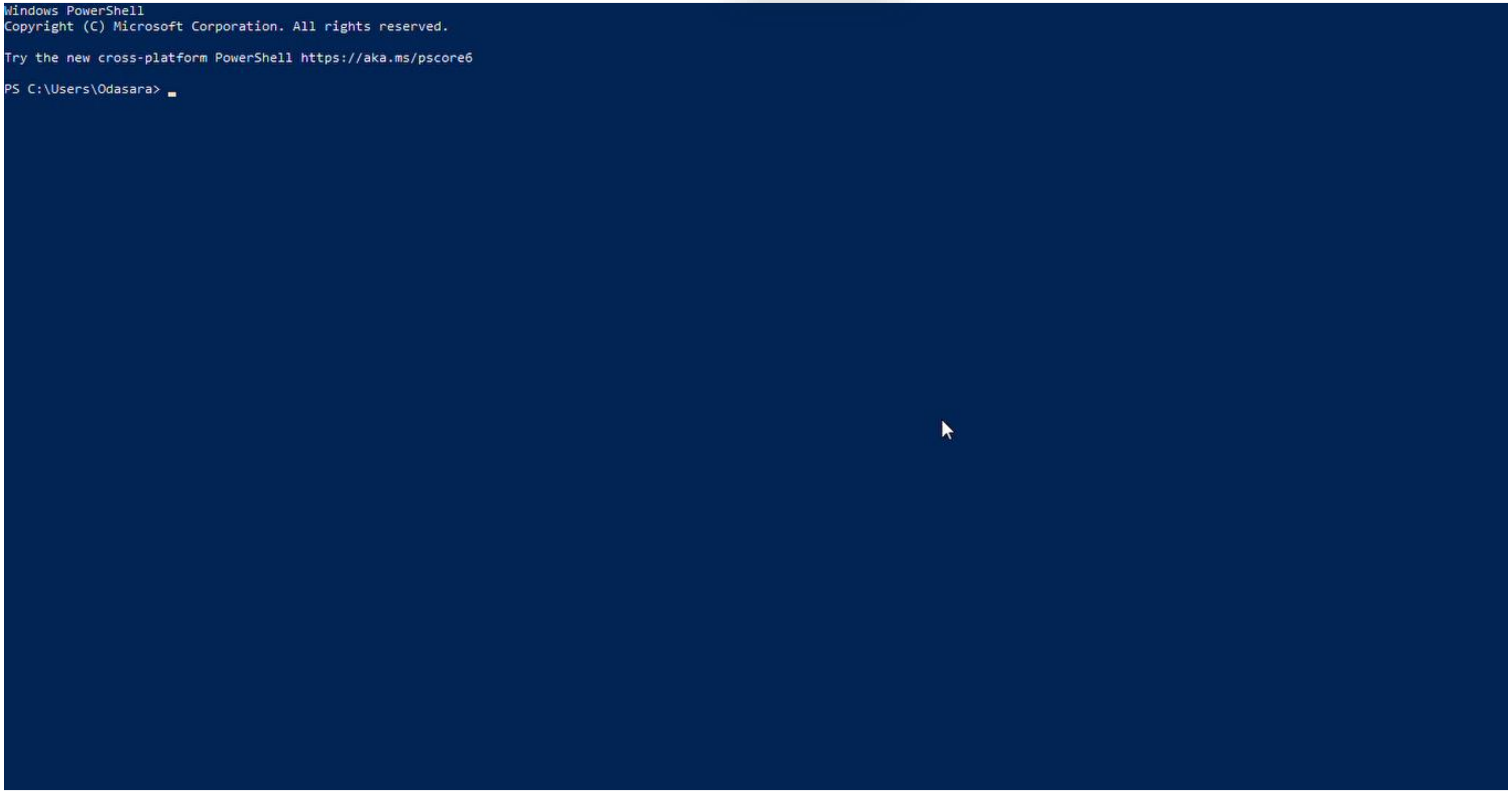
Enhanced User Experience

- Simple design
- Save login credentials
- Dealing with forgotten passwords
- Send notifications
- Easy access for frequently used recipes

Demonstration



Back end



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Odasara> _
```

Functionalities

Security

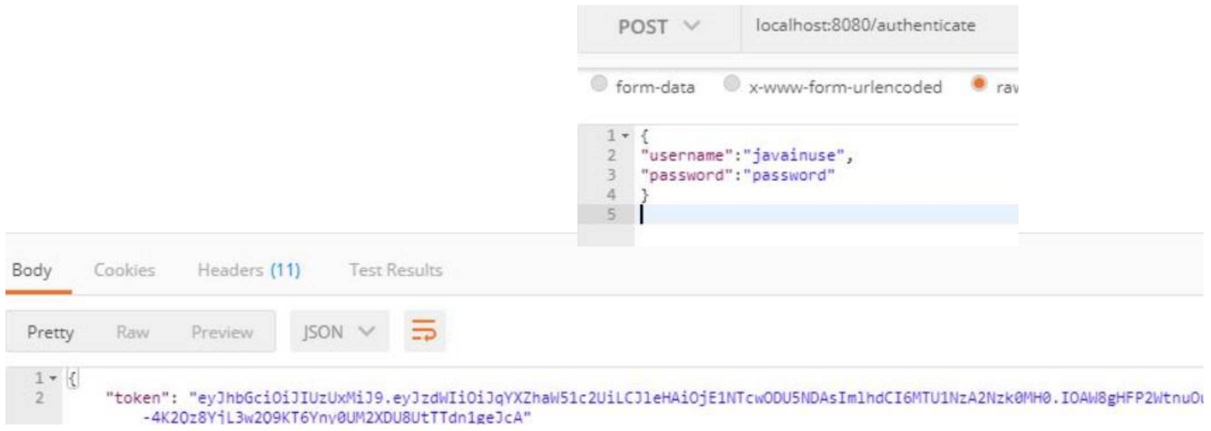
- Password Encoding Using BCrypt
- Use of JWT authentication

Reliability

- Well-secured Features

Scalability

- Use of smaller, independent packages or modules while coding


POST localhost:8080/authenticate

form-data x-www-form-urlencoded raw

```
{
"username":"javainuse",
"password":"password"
}
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview JSON

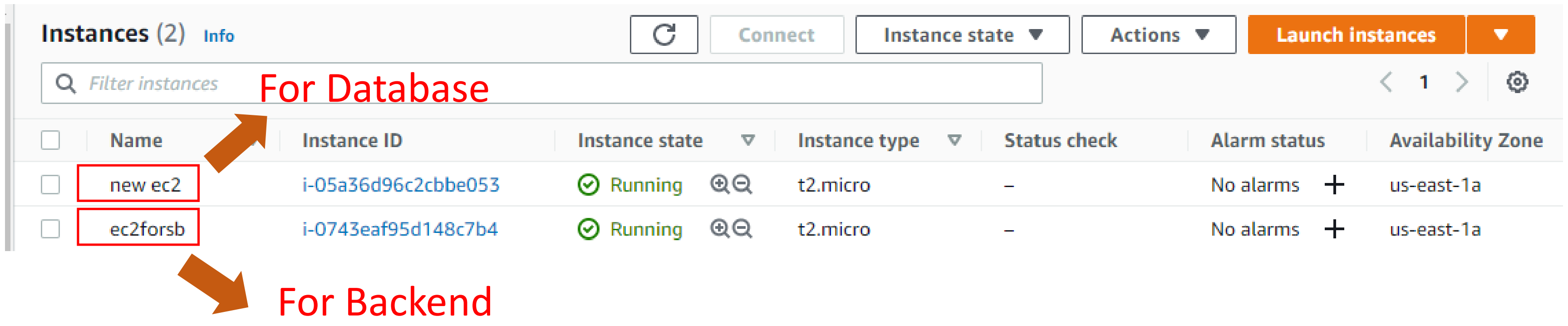
```
{
    "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJqYXZhaW51c2UiLCJleHAiOjE1NTcwODU5NDAsImlhdCI6MTU1NzA2Nzk0MH0.IOAW8gHFP2WtnuO
      -4K2Oz8YjL3w2O9KT6Yny0UM2XDU8UtTTdn1geJcA"
```

Cloud Deployment

Cloud deployment

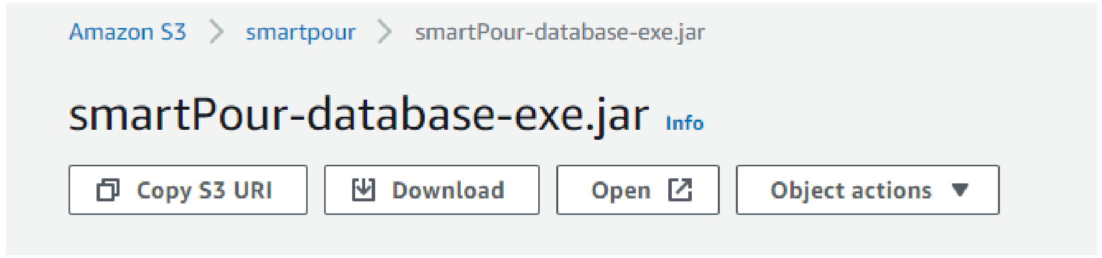
Steps

- Created EC2 instances with required apps



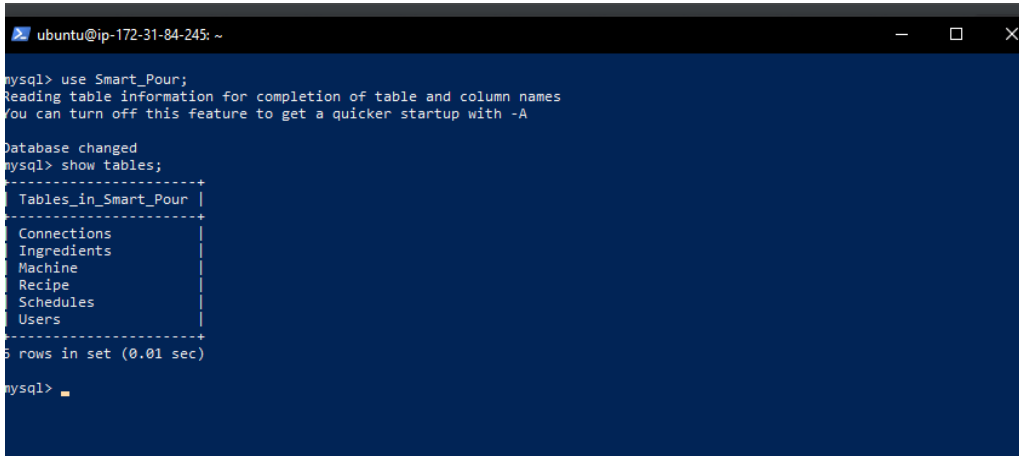
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
	new ec2	i-05a36d96c2cbbe053	Running	t2.micro	–	No alarms	us-east-1a
	ec2forsb	i-0743eaf95d148c7b4	Running	t2.micro	–	No alarms	us-east-1a

- Created a S3 bucket and stored the backend application


Amazon S3 > smartpour > smartPour-database-exe.jar

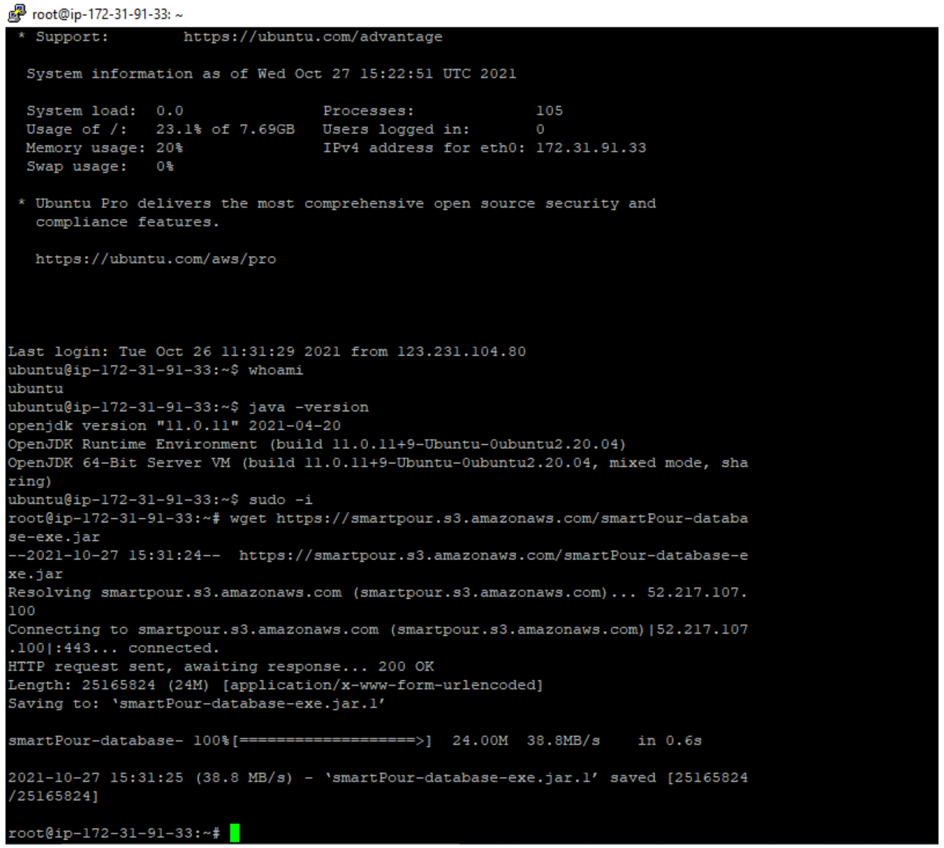
smartPour-database-exe.jar Info

- Deployed complete database and the backend on AWS EC2



ubuntu@ip-172-31-84-245: ~

```
ysql> use Smart_Pour;
eading table information for completion of table and column names
ou can turn off this feature to get a quicker startup with -A

atabase changed
ysql> show tables;
---------------------+
 Tables_in_Smart_Pour |
---------------------+
 Connections          |
 Ingredients          |
 Machine              |
 Recipe               |
 Schedules            |
 Users                |
---------------------+
 rows in set (0.01 sec)

ysql> 
```



root@ip-172-31-91-33: ~

```
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Oct 27 15:22:51 UTC 2021

  System load:  0.0               Processes:             105
  Usage of /:   23.1% of 7.69GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for eth0: 172.31.91.33
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro



Last login: Tue Oct 26 11:31:29 2021 from 123.231.104.80
ubuntu@ip-172-31-91-33:~$ whoami
ubuntu
ubuntu@ip-172-31-91-33:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sha
ring)
ubuntu@ip-172-31-91-33:~$ sudo -i
root@ip-172-31-91-33:~# wget https://smartpour.s3.amazonaws.com/smartPour-databa
se-exe.jar
--2021-10-27 15:31:24--  https://smartpour.s3.amazonaws.com/smartPour-database-e
xe.jar
Resolving smartpour.s3.amazonaws.com (smartpour.s3.amazonaws.com)... 52.217.107.
100
Connecting to smartpour.s3.amazonaws.com (smartpour.s3.amazonaws.com)|52.217.107
.100|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25165824 (24M) [application/x-www-form-urlencoded]
Saving to: ‘smartPour-database-exe.jar.1’

smartPour-database- 100%[===================>]  24.00M  38.8MB/s    in 0.6s

2021-10-27 15:31:25 (38.8 MB/s) - ‘smartPour-database-exe.jar.1’ saved [25165824
/25165824]

root@ip-172-31-91-33:~# 
```

Functionalities

Security
- Security groups
- S3 security
- Access Management Control

Scalability

•Auto-scaling feature
•Automatically maintain predictable performance at the lowest possible cost.

Reliability
- Backups of the database provided by AWS

Testing

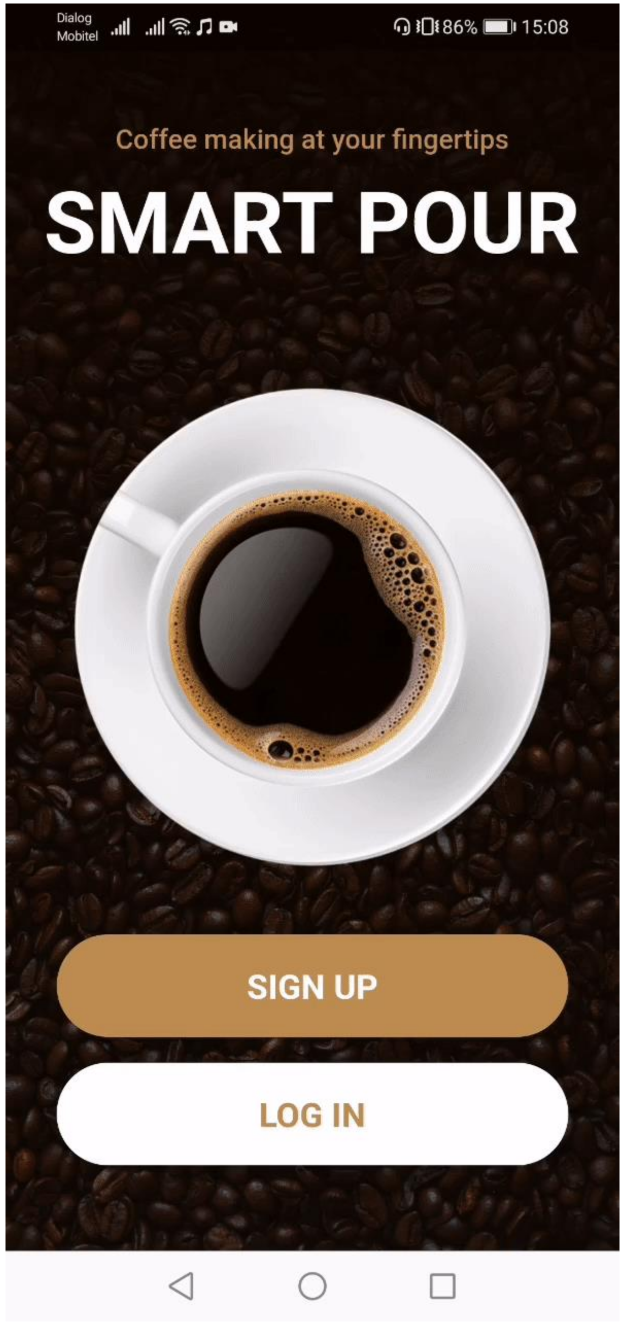
Validation

WHY?

- Tables in database should be correctly updated
- To make sure that the system is only accessible by users with accurate details
- To avoid spams

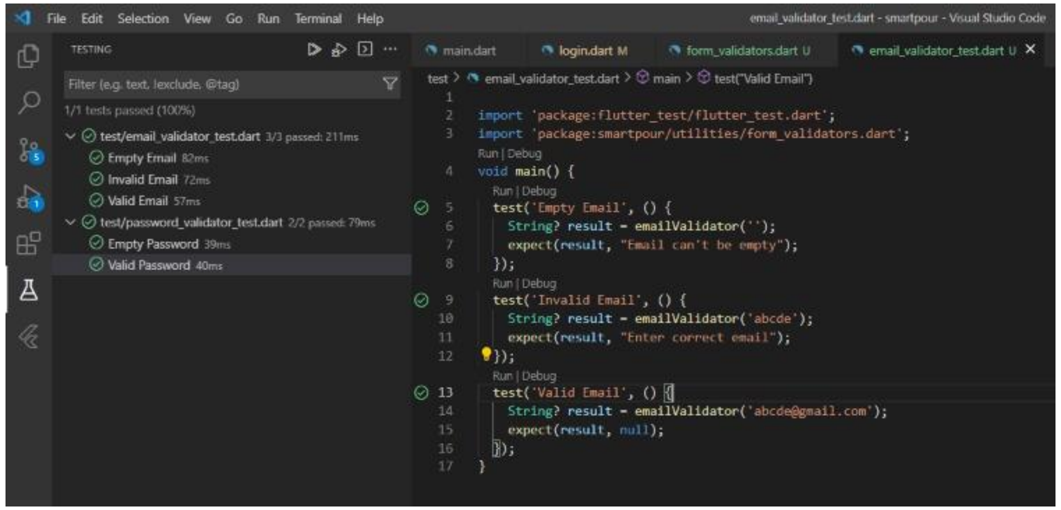
TESTED

- Tested for empty fields
- Correct form of Email



Validation Testing – Unit testing

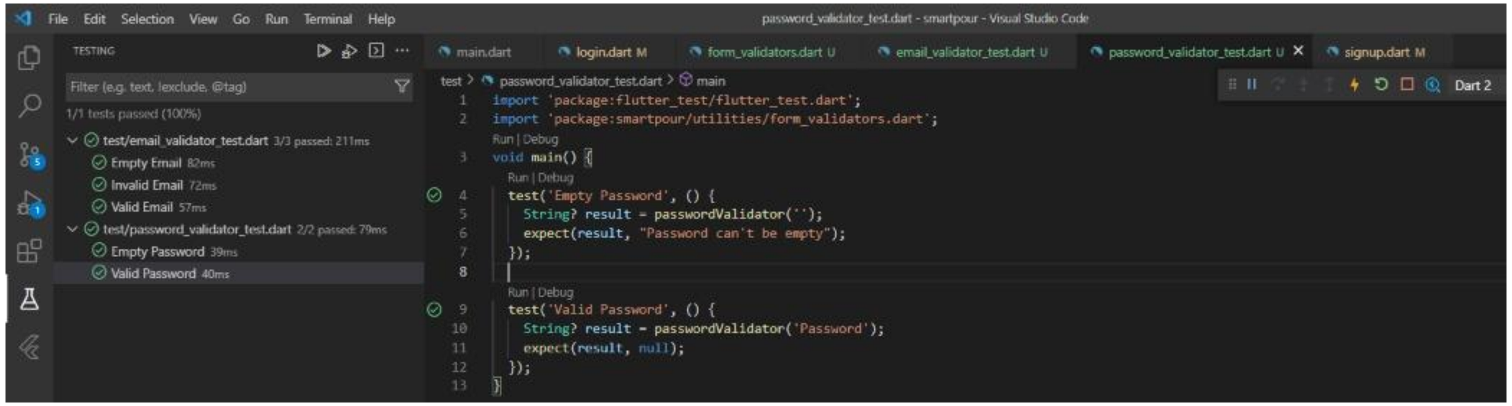
Email Validation



```
TESTING
1/1 tests passed (100%)
test/email_validator_test.dart 3/3 passed: 211ms
  Empty Email 82ms
  Invalid Email 72ms
  Valid Email 57ms
test/password_validator_test.dart 2/2 passed: 79ms
  Empty Password 39ms
  Valid Password 40ms
```

```dart

import 'package:flutter_test/flutter_test.dart';
import 'package:smartpour/utilities/form_validators.dart';
void main() {
  test('Empty Email', () {
    String? result = emailValidator('');
    expect(result, "Email can't be empty");
  });
  test('Invalid Email', () {
    String? result = emailValidator('abcde');
    expect(result, "Enter correct email");
  });
  test('Valid Email', () {
    String? result = emailValidator('abcde@gmail.com');
    expect(result, null);
  });
}
```

Password Validation



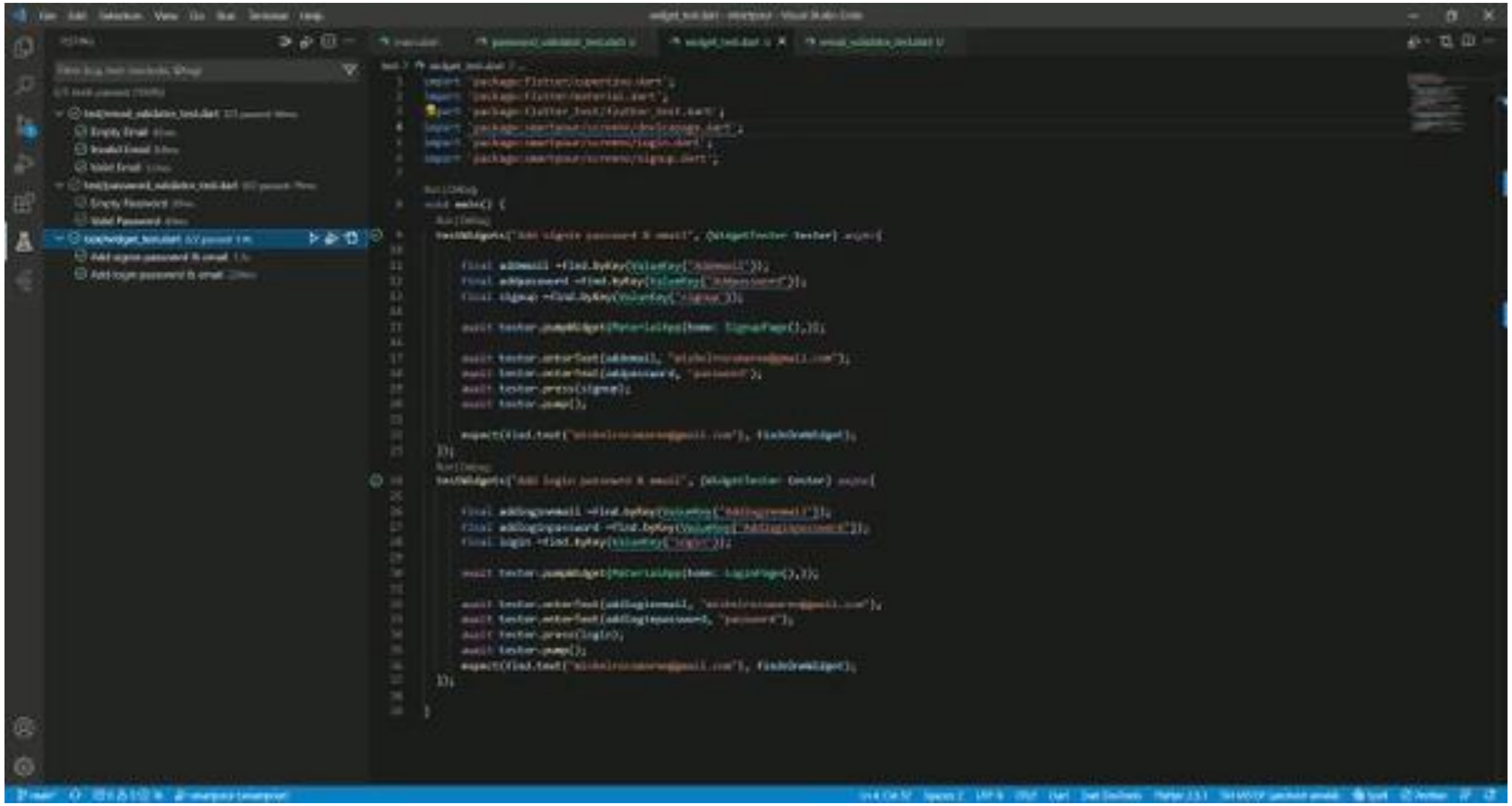
```
TESTING
1/1 tests passed (100%)
test/email_validator_test.dart 3/3 passed: 211ms
  Empty Email 82ms
  Invalid Email 72ms
  Valid Email 57ms
test/password_validator_test.dart 2/2 passed: 79ms
  Empty Password 39ms
  Valid Password 40ms
```

```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:smartpour/utilities/form_validators.dart';
void main() {
  test('Empty Password', () {
    String? result = passwordValidator('');
    expect(result, "Password can't be empty");
  });

  test('Valid Password', () {
    String? result = passwordValidator('Password');
    expect(result, null);
  });
}
```

Widget Testing

WHY?

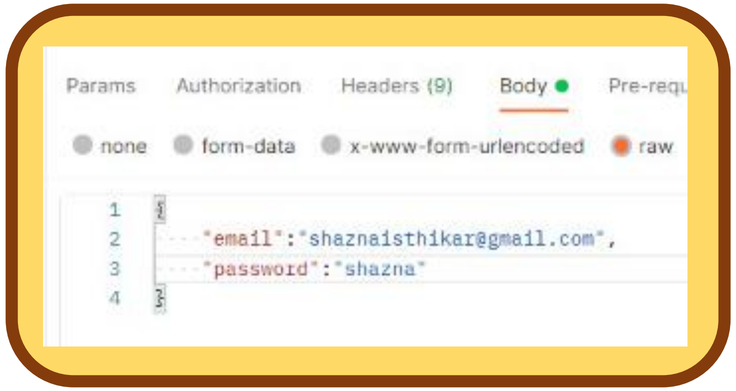
- To verify that the widget's UI looks and interacts as expected



```
test/email_validator_test.dart
  Empty Email
  Invalid Email
  Valid Email
test/password_validator_test.dart
  Empty Password
  Valid Password
test/widget_test.dart
  Add signin password & email
  Add login password & email
```

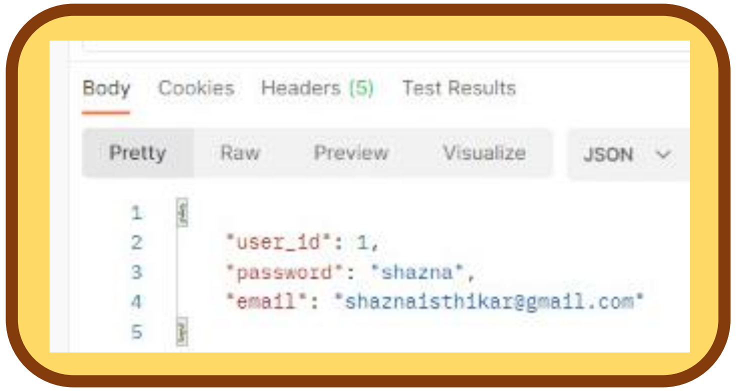
```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:smartpour/screens/[illegible].dart';
import 'package:smartpour/screens/login.dart';
import 'package:smartpour/screens/signup.dart';

void main() {
  testWidgets('Add signin password & email', (WidgetTester tester) async{

      final addemail =find.byKey(ValueKey('Addemail'));
      final addpassword =find.byKey(ValueKey('Addpassword'));
      final signup =find.byKey(ValueKey('signup'));

      await tester.pumpWidget(MaterialApp(home: SignupPage(),));

      await tester.enterText(addemail, '[illegible]@gmail.com');
      await tester.enterText(addpassword, 'password');
      await tester.press(signup);
      await tester.pump();

      expect(find.text('[illegible]@gmail.com'), findsOneWidget);
  });
  testWidgets('Add login password & email', (WidgetTester tester) async{

      final addloginemail =find.byKey(ValueKey('Addloginemail'));
      final addloginpassword =find.byKey(ValueKey('Addloginpassword'));
      final login =find.byKey(ValueKey('login'));

      await tester.pumpWidget(MaterialApp(home: LoginPage(),));

      await tester.enterText(addloginemail, '[illegible]@gmail.com');
      await tester.enterText(addloginpassword, 'password');
      await tester.press(login);
      await tester.pump();
      expect(find.text('[illegible]@gmail.com'), findsOneWidget);
  });

}
```

API Testing – Unit testing



```
{
    "email":"shaznaisthikar@gmail.com",
    "password":"shazna"
}
```

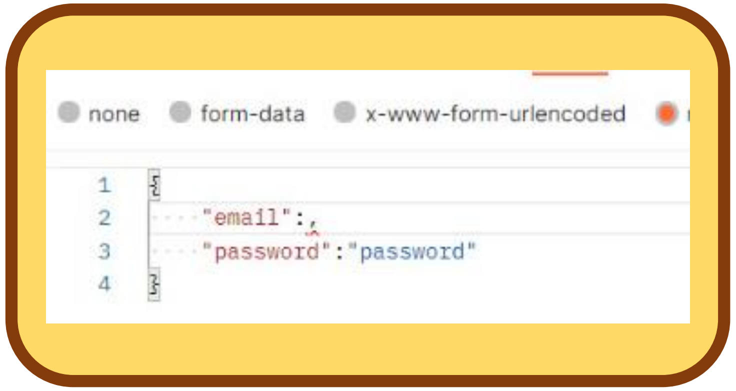
Email ✓
Password ✓



```
{
    "user_id": 1,
    "password": "shazna",
    "email": "shaznaisthikar@gmail.com"
}
```

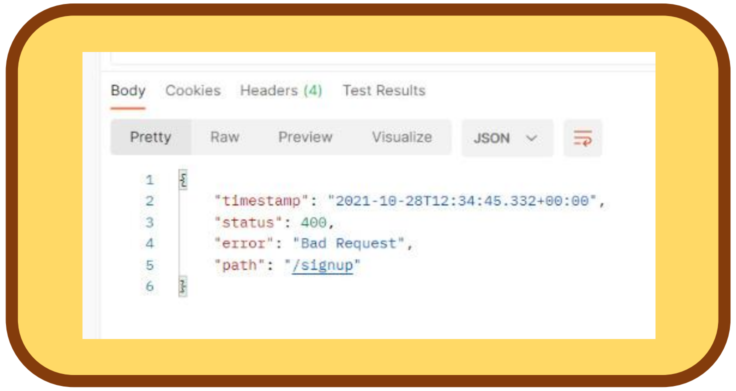
Request Panel

Reply Panel



```
{
    "email":,
    "password":"password"
}
```

Email ✗
Password ✓



```
{
    "timestamp": "2021-10-28T12:34:45.332+00:00",
    "status": 400,
    "error": "Bad Request",
    "path": "/signup"
}
```

Request Panel

Reply Panel

Q
A

THANK
YOU !

Functional completeness
● Completeness of back-end software
● Completeness of front-end software
● Cloud deployment
● Designs for embedded node hardware
Understanding about the system
● Ability to provide a clear overview of the system
● Ability to clearly explain features and functionalities (including reliability, scalability and security aspects)
● Ability to clearly explain implementation details
User Experience
● Attention paid to enhance UX of software/hardware components and of the overall product
Software Testing
● Details of three or more tests carried out on the software components (what was tested?, why was it tested?, how was the test done?, results and findings