National University of Computer and Emerging Sciences - FAST Computer Science Department

Fundaments of Software Engineering
CS-2004

Submitted to:

Ma'am Maheen Arshad

Submitted by:

1. Musaab Imran (20i-1794)
2. Muhammad Ismail Ramzan (20i-1941)
3. Muhammad Usman Shahid (20i-1797)

Contents

<u>**Fast Bus Management System**</u>

The module we will be covering in sprint 1 contains three user stories. All three user stories are interlinked. The user stories are related to login, registration, and log-out. We will be fully implementing all three of the user stories in sprint 1 which would include both the front and back end.

User Stories:

01

ID: US-01   **Title: User Login**

- **Created by:** Group
- **Date created:** 01/03/2022
- **Last updated by:**24/03/2022

As a customer, I want to have proper login for different users, including students, drivers, and admin. After the log in each user should see his/her respective page.

Description:

- Users will have to log in.
- Users would log in through their username and password.
- There must be a check on the passwords.

Acceptance criteria:

- Check for new users to make their accounts and add the information to the database.
- If the user is already registered then allow him/her to enter.

Priority:

High

Estimate hours:

7-8

06

ID: US-06 **Title: User Registration**

- **Created by:** Group
- **Date created:** 01/03/2022
- **Last updated by:**04/03/2022

As a customer, I want the user (students/drivers) to register if he/she doesn't have an account already. The registration must be done through the below fields.

Description:

Things to be asked:

- Name
- Username
- Password (strong check).
- Phone number

Acceptance criteria:

- The password should be strong.
- All the data entered in the fields must be right.
- Different checks on all the fields asked in the registration section.

Priority:

High

Estimate hours:

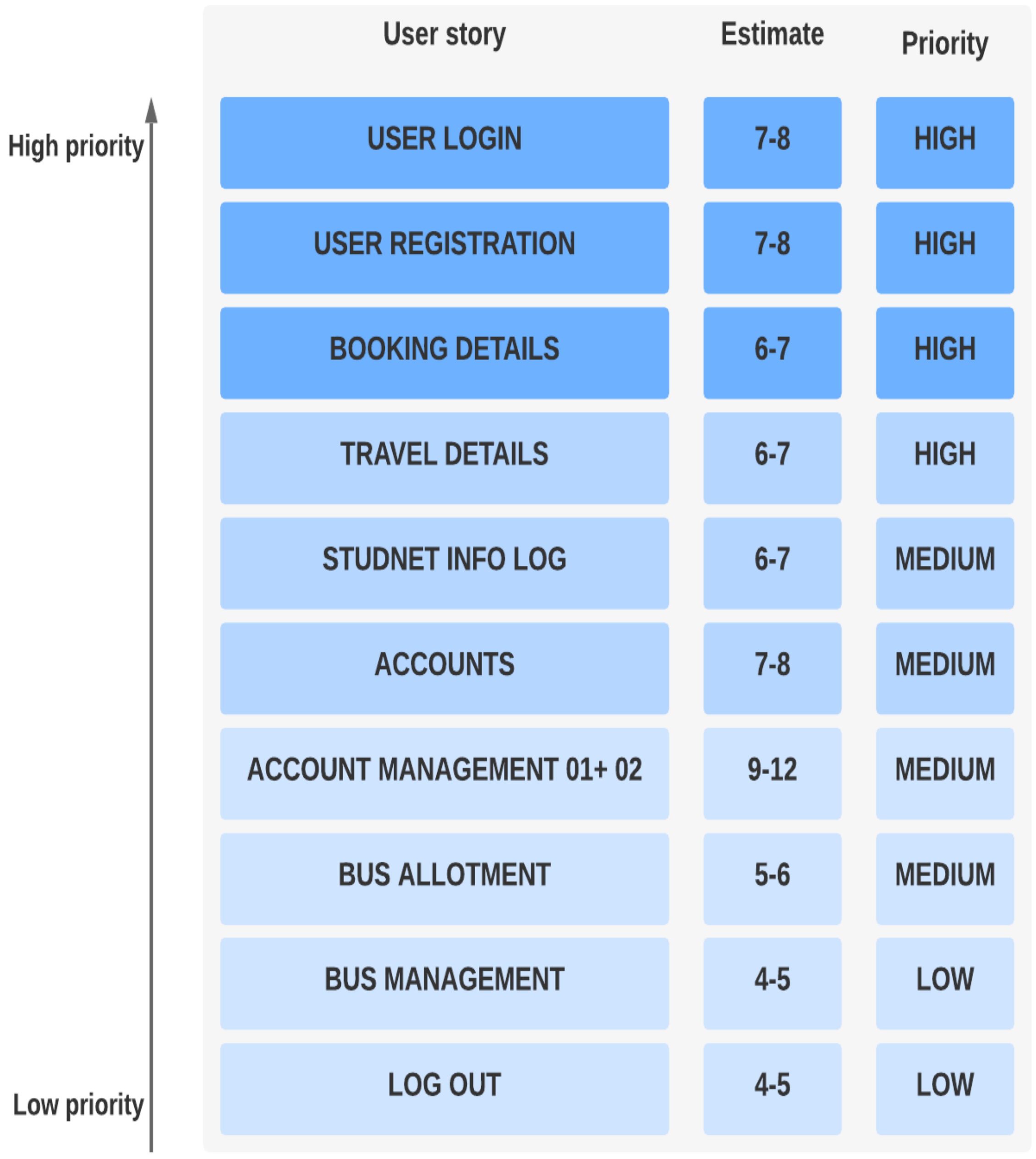
7-8

ID: US-11 **Title: Log-out** ➢ **Created by:** Group ➢ **Date created:** 01/03/2022 ➢ **Last updated by:**04/03/2022 As a customer, I want that that whenever the user has done his/her work their account should have a proper logout to have a sense of security. So, their accounts couldn't be accessed by others. **Description:** The user while using the app should be able to sign out or log out. The log-out should be proper and if the user wants to use the app again then he/she should log in again. **Acceptance criteria:** ○ If the user clicks logout, he/she won't be able to remain logged in, if they want to use the app again, they must log in again.	**Priority:** High **Estimate hours:** 4-5

Project Backlog

USER STORIES	DESCRIPTION
US01:	Login system for students, bus drivers, and admin.
US02:	Chalan status for students and salary confirmation for the bus drivers.
US03:	Seat booking and boarding number generation for students.
US04:	Travel information, details like bus and driver numbers, and time mentioned routes on maps.
US05:	Passenger information log, showing names, tokens, and picking points to the driver.
US06:	Students and drivers register who are traveling for the first time against different information.
US07:	Daily bus and route allotment by the admin.
US08:	Account deletion of students and drivers who have left.
US09:	And adding new students and drivers, also creating their accounts.
US10:	Maintaining buses data, also adding and removing the buses.
US11:	Logout of the account after the user enters the logout button.

PRODUCT BACKLOG


High priority
Low priority

User story	Estimate	Priority
USER LOGIN	7-8	HIGH
USER REGISTRATION	7-8	HIGH
BOOKING DETAILS	6-7	HIGH
TRAVEL DETAILS	6-7	HIGH
STUDNET INFO LOG	6-7	MEDIUM
ACCOUNTS	7-8	MEDIUM
ACCOUNT MANAGEMENT 01+ 02	9-12	MEDIUM
BUS ALLOTMENT	5-6	MEDIUM
BUS MANAGEMENT	4-5	LOW
LOG OUT	4-5	LOW

Sprint Backlog

Sprint 1 (1.5 weeks):
US01- User Login
US06- User Registration
US11- Logout

Sprint 2 (1weeks):
US03- Booking Details
US04- Travel Details

Sprint 3 (1 week):
US08- Account Management 01
US09- Account Management 02

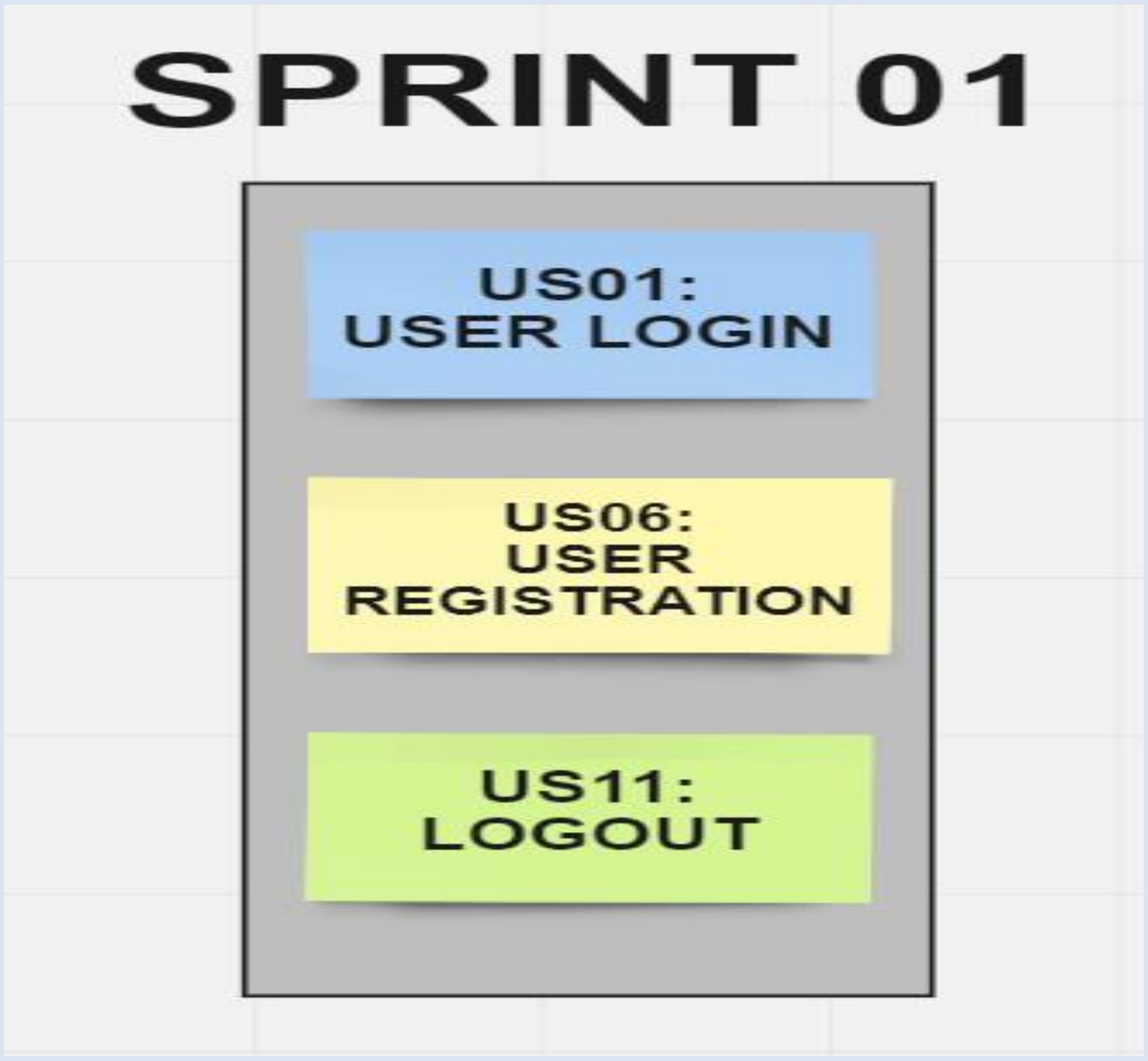
Sprint 4 (1 week):
US07- Bus Allotment
US10- Bus Management

Sprint 5 (1 weeks):
US05- Student Info Log
US02- Accounts(finance)

- The main goal is to complete the project in approximately 2.5 months.
- Each sprint contains interlinked modules for fast work.
- The average maximum time for each sprint is 7-8 days.

Sprint 01:

Requirement#	Description
1	Login system for students, bus drivers, and admin.
6	Registration for students and drivers.
11	Logout of the account after the user enters the logout button.



Tasks:

US01: User login (Driver, Admin, Student)	**Full Front-End Implementation:** **T101-** Made a widget having different quantities. **T102 –** Inserted the image of the logo of our app. **T103 –** Aligned the image according to the page. **T104 -** Inserted the text widget to write the user's name and stored the output in the variable. **T105-** Inserted the text widget to write the password and stored the output in the variable. **T106-** Set the alignment of both widgets. **T107-** then called the flat button class and called the layout widgets which are child and children. **T108-** called the text widget and wrote register. **T109-** by clicking this we enter the page where we can register if we don't have the account. **T110-** Made the check from the password by applying loops and checking for**:** **1.** Special characters **2.** Both upper and lower case **3.** Numbers **4.** Equal or more than the 8-character length These were the checks for password **T111-** made user name check where the user can't enter the special characters to avoid any injection attack (SQL). **T112-** the login page was the same for all the users. Check for different users was done in the backend implementation of this task. **Full Back-End Implementation:** **T101-** Created the Authentication into the Firebase **T102 –** Connected the Firebase to the Flutter application **T103 –** Saved the data of user registration into the flutter Authentication service

	T104 - Showed Different errors based on input provided by the user in case of invalid input **T105 –** Database created to save data based on different attributes of the user **T106 –** Made Fire store Datastore configuration **T107 –** Saved the registered data into the database **T108 –** Verify the login credentials from the server auth service **T109 -** User Error handling in case of wrong credentials **T110 –** After logging in, open the right home screen by fetching the data from the database.
US01 + US06: Admin Registration & Login	**Full Front-End Implementation:** **T1+601-** Hardcode the credentials of the admin at the backend no registration front end for admin **T1+602 –** Admin login by entering credentials **T1+603 –** Made the check from the password by applying loops and checking for**:** **5.** Special characters **6.** Both upper and lower case **7.** Numbers **8.** Equal or more than the 8-character length **T1+603 –** made user name check where the user can't enter the special characters to avoid any injection attack (SQL). **T1+605 –** the login page was the same for all the users. Check for different users was done in the backend implementation of this task **T1+0606-** mostly many things in task 01 & task 06 match this T1+T06

	Full Back-End Implementation: **T1+601 –** Added Credential to the Firebase Auth service **T1+602 –** Added the Admin data into the Fire store Database **T1+602 –** As the user click login, credentials are checked against the admin **T1+603 –** In the case of matching credentials, the data of the admin is fetched from the Fire store database. **T1+605 –** Based on that Fire store database the admin route is called and is shown to the user
US06: User Registration (Driver And Student)	**Full Front-End Implementation:** **T601-** Made a widget having different quantities. **T602 –** Inserted the bar on top and the option to go back. **T603 –** Inserted the form widget and then the text fields. **T604 -** Inserted the text widget to write the user's name and stored the output in the variable. **T605-** Inserted the text widget to write the password and stored the output in the variable. **T606-** Inserted the text widget to write the username and stored the output in the variable. **T607-** Inserted the text widget to write the phone number and stored the output in the variable. **T608-** Added a dropdown for the selection of driver and student to register accordingly **T609-** Added a checkbox so to ensure the user accepts our privacy policy **T610-** Made the check from the password by applying loops and checking for**:** • Special characters • Both upper and lower case • Numbers • Equal or more than the 8-character length

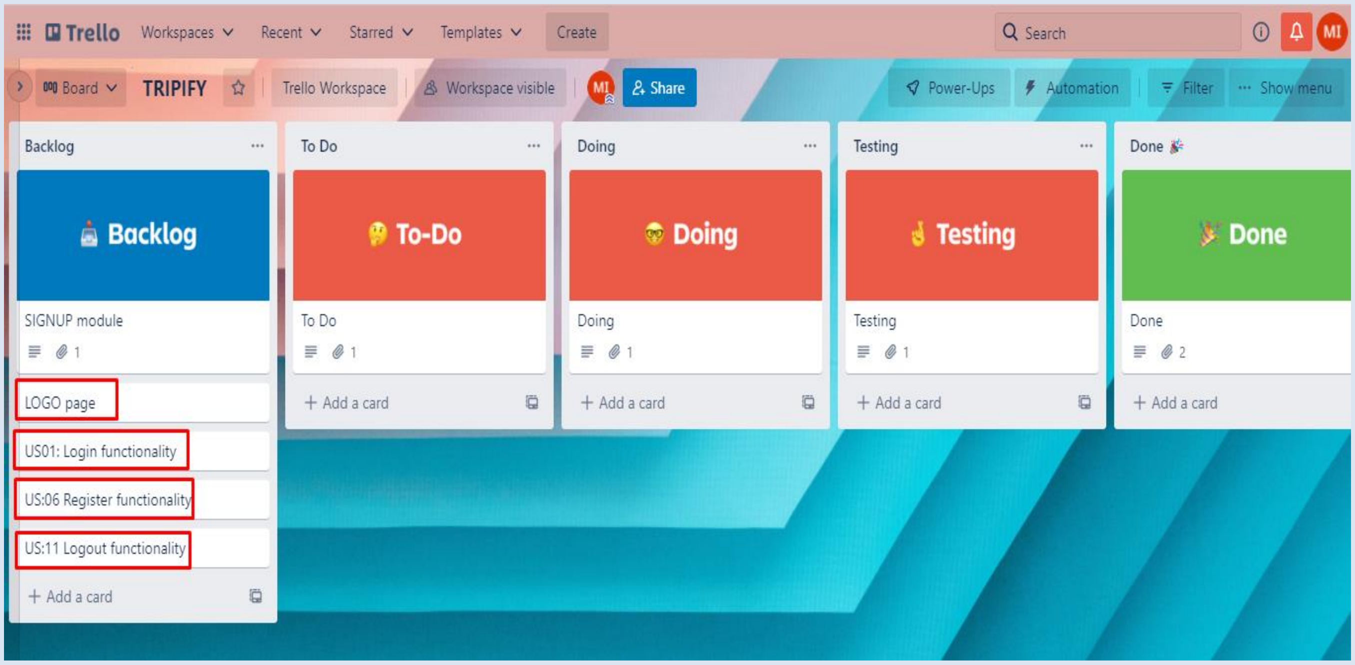
	These were the checks for password **T611-** made other text field checks where the user can't enter the special characters to avoid any injection attack (SQL). **T612-** the login page was the same for all the users. Check for different users was done in the backend implementation of this task. **T613-** added the elevated button and then register the user in the backend. If registered successfully then is redirected to the login page. **Full Back-End Implementation:** **T601-** Created the Authentication into the Firebase for the driver and student **T602** – After Successful auth, the data of that user is saved at the Firebase Auth service and the Fire store database. **T603** – The user can select whether driver or student in the registration phase **T604** – In case driver "2" is stored in the Database **T605** – In case student "1" is stored in the Database **T606** – Auth service is used to handle the registration **T607**- Validation against the same username and other errors is applied in the code
US11: **Logout**	**Full Front-End Implementation:** **T11/01-** Made a widget having different quantities. **T11/02 –** Inserted the image of the logo of our app. **T11/03 –** Aligned the image according to the page. **T11/04 –** make the lower navigation bar on the home page give the option to log out. **T11/05-** added the logout icon also the label to make user friendly. **T11/06-** Set the alignment of both widgets. **T11/07-** designing the navigation bar by highlighting the current one.

	T11/08- when logout is clicked gone to the backend and implemented the logout then **T11/09-** after logout pressed and done used the navigator method to navigate to the home page. **T11/10-**the home page logo one displays to start again, another registration or logout **Full Back-End Implementation:** **T11/01-** A global variable is used to detect the current state of the user. **T11/02-** if the variable is true then the user is logged in and if the variable is false then the user is logged out. **T11/03-** This variable is made via stream so it can run in the background all the time so we can detect whether the user is logged in or not. **T11/04-** The user is shown with the logout button. Once that button is pressed a stream detects that and the state of the global variable is set to false (logged out) **T11/05-** After this action, the user is no more logged in so is directed to the login route.

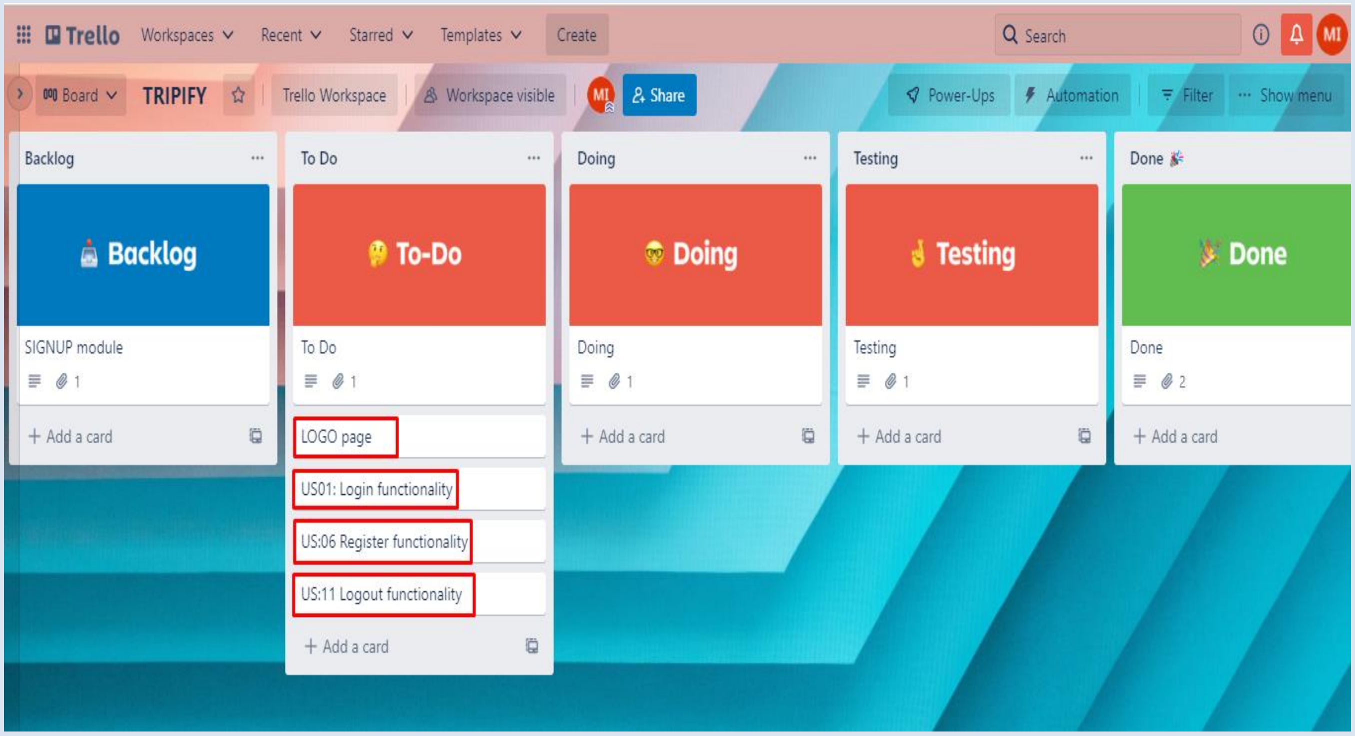
LOGO page Implementation	**T01-** Made a widget having different quantities. **T02 –** Inserted the images of the logo. **T03 –** made an elevated button so to go next to the login interface. **T04 –** added below the IMU solutions powered text. **T05-** decided on the animation to be used and decided to use the explicit animations. **T06-** for explicit animation using the animator builder widget. **T07-** then gone for the transform. Translate to move horizontally. **T08-** then said to be running forever **T09-** when the user clicks the button let's go, then displays the login interface

Trello:

- At the start of the project:

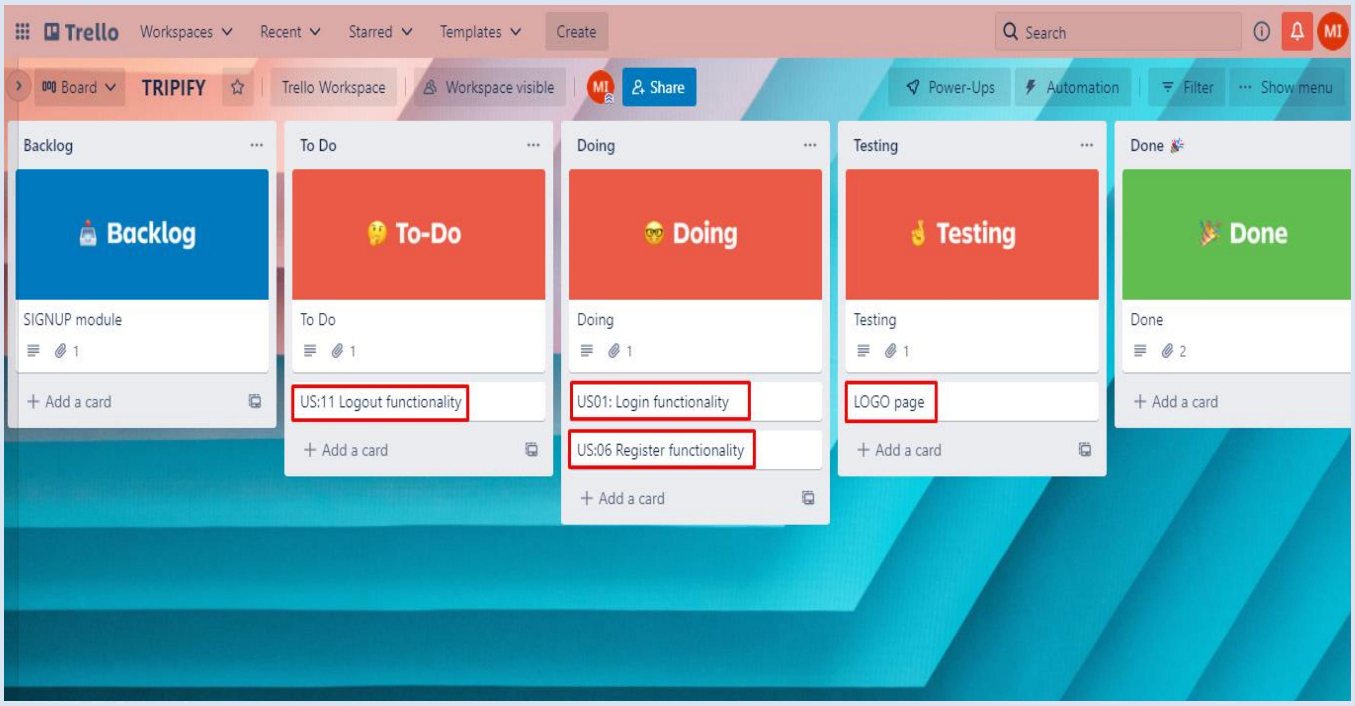


Backlog	To Do	Doing	Testing	Done
Backlog	To-Do	Doing	Testing	Done
SIGNUP module	To Do	Doing	Testing	Done
LOGO page				
US01: Login functionality				
US:06 Register functionality				
US:11 Logout functionality				
+ Add a card	+ Add a card	+ Add a card	+ Add a card	+ Add a card

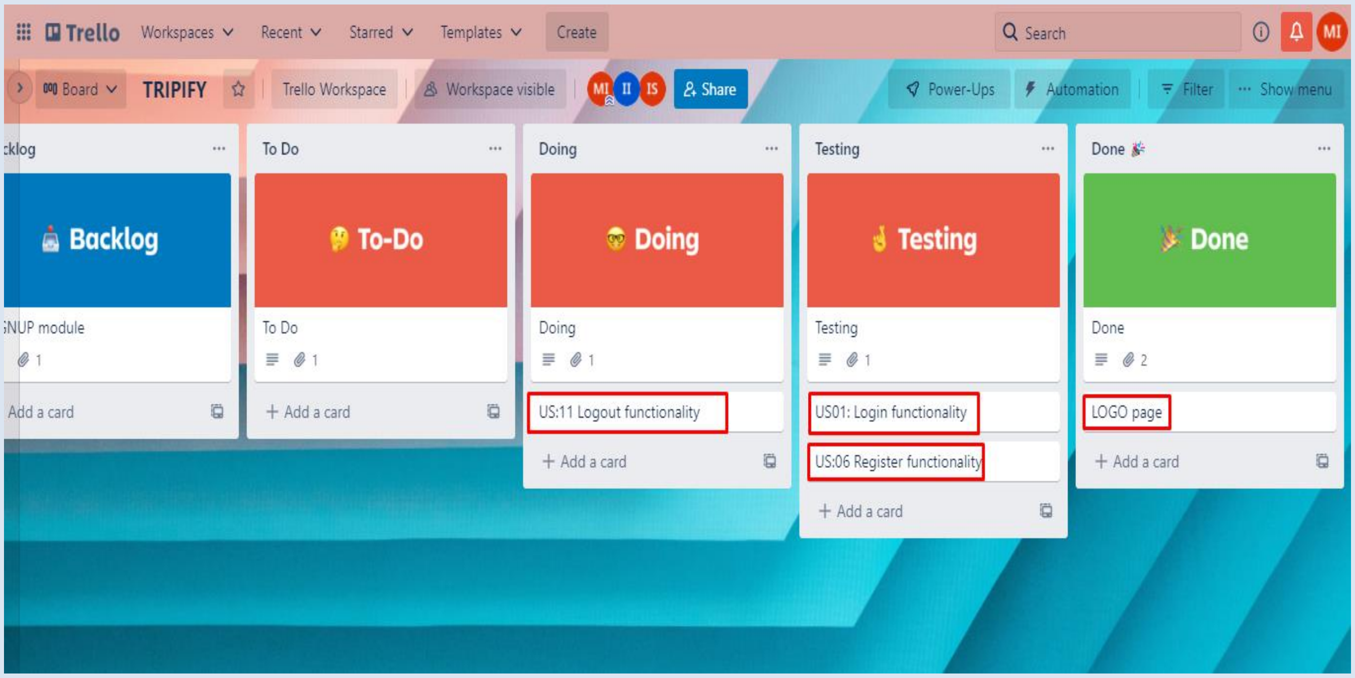


Backlog	To Do	Doing	Testing	Done
Backlog	To-Do	Doing	Testing	Done
SIGNUP module	To Do	Doing	Testing	Done
+ Add a card	LOGO page	+ Add a card	+ Add a card	+ Add a card
	US01: Login functionality			
	US:06 Register functionality			
	US:11 Logout functionality			
	+ Add a card			

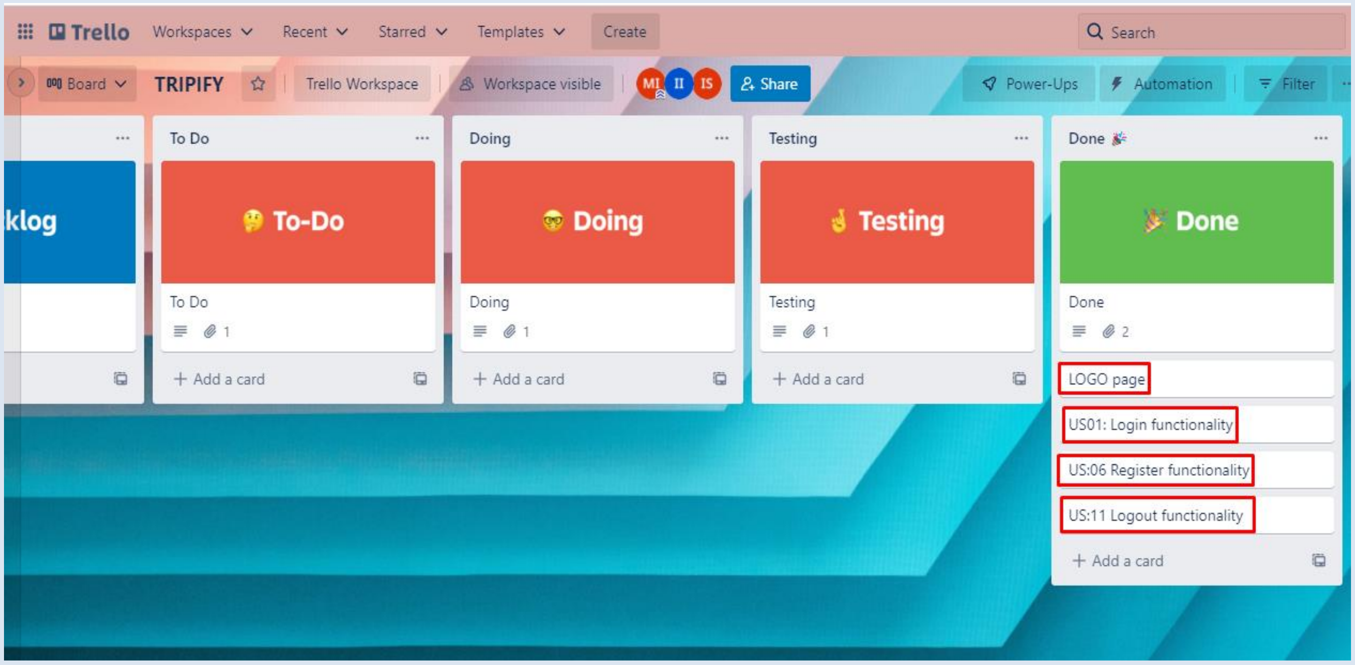
- At the middle of the project:



- When 80% of the project was completed:



- At the end of the project:



To Do

Doing

Testing

Done

LOGO page

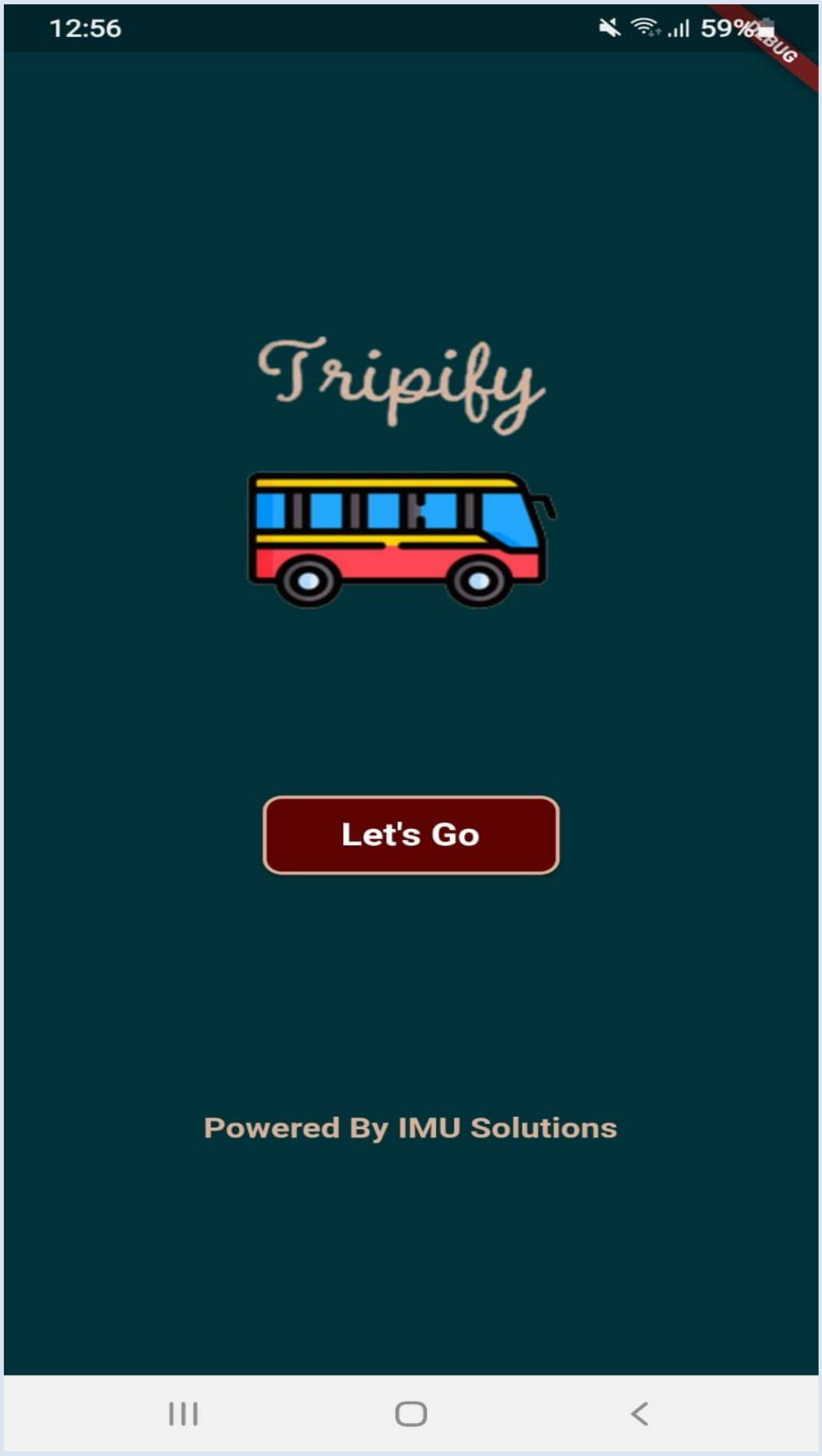
US01: Login functionality

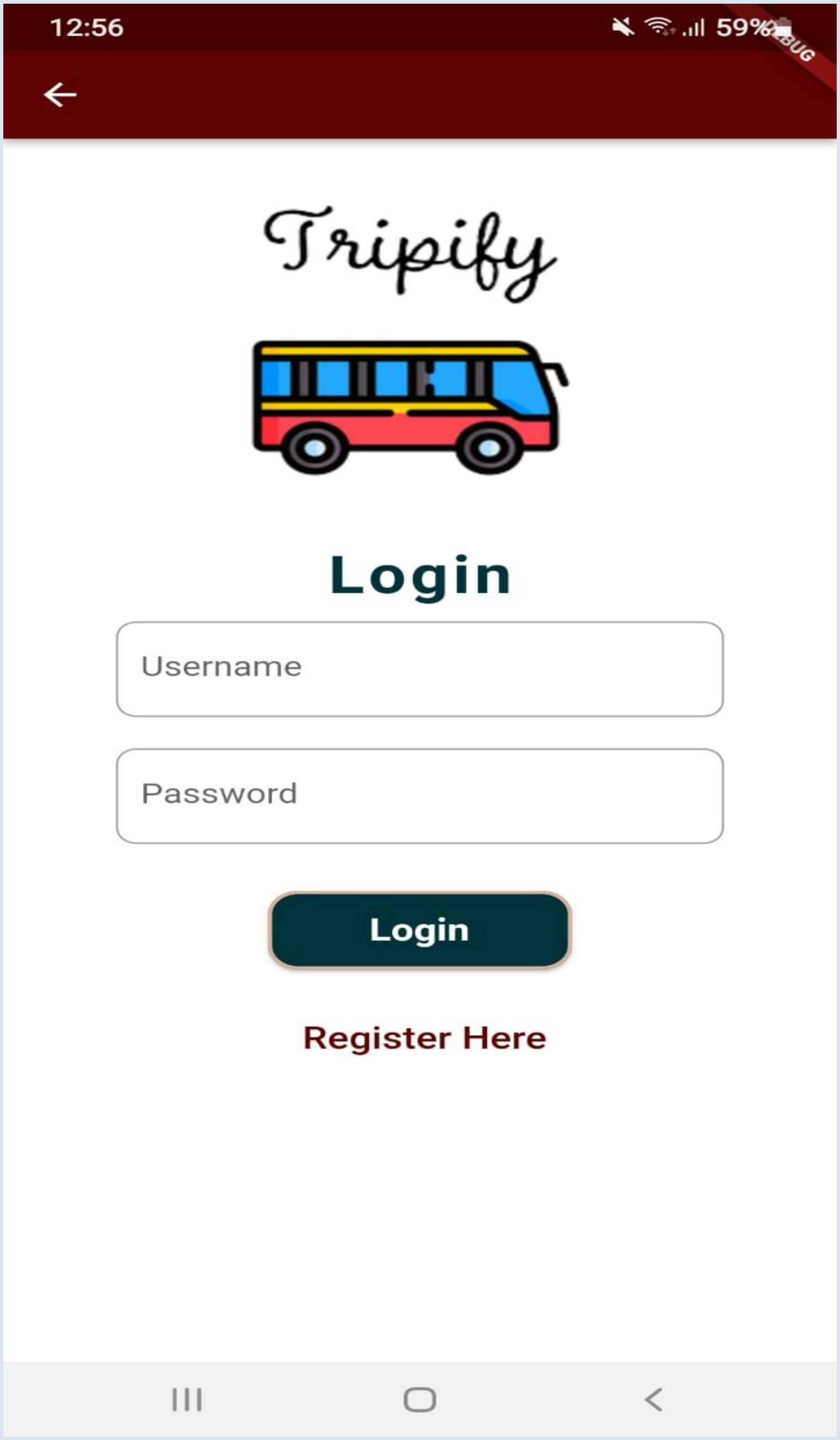
US:06 Register functionality

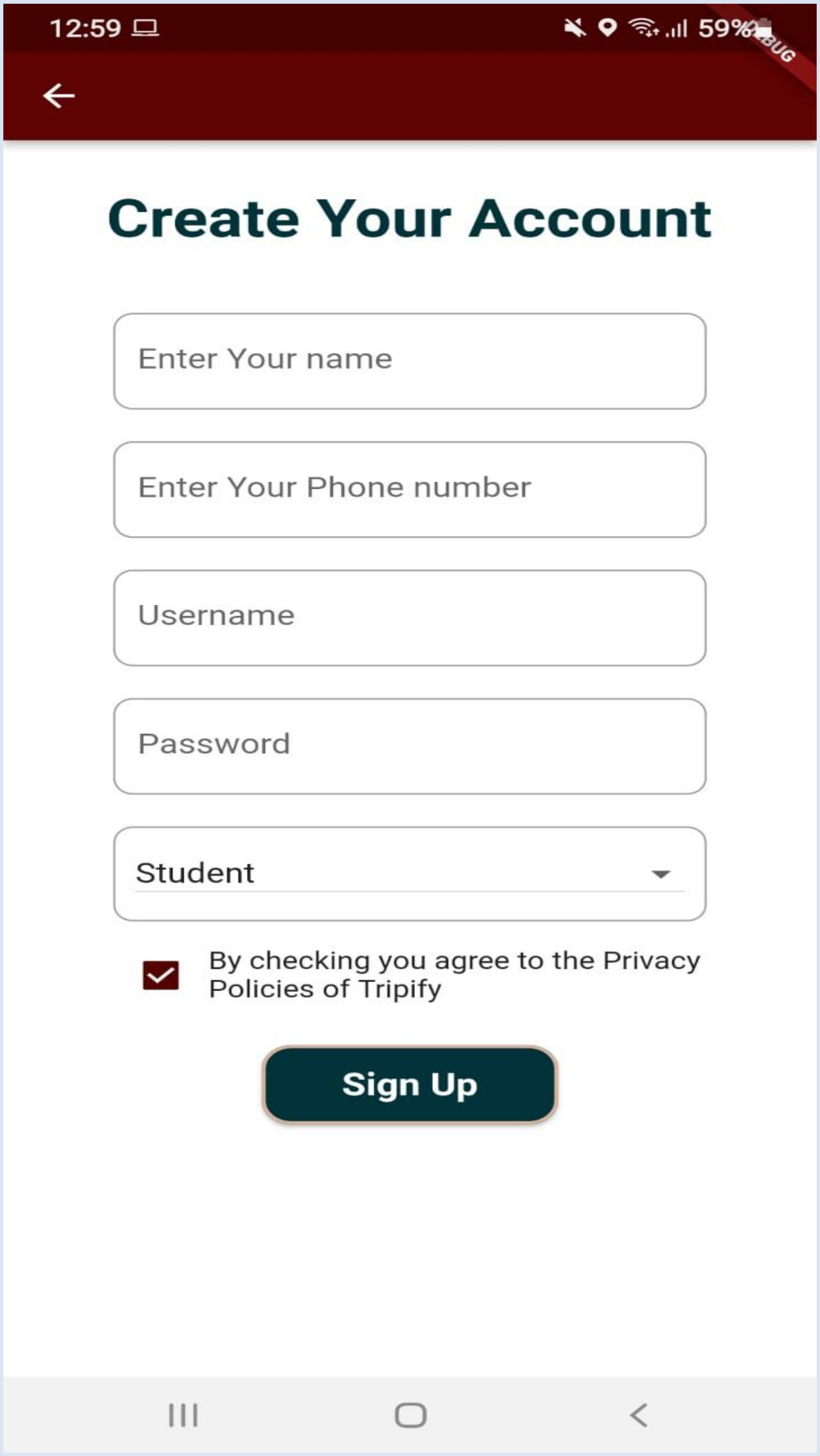
US:11 Logout functionality

Front-end implementation:

Main Pages:


12:56
59%
DEBUG
Tripify
Let's Go
Powered By IMU Solutions


12:56
59%
DEBUG
Tripify
Login
Username
Password
Login
Register Here


12:59 59%

Create Your Account

Enter Your name

Enter Your Phone number

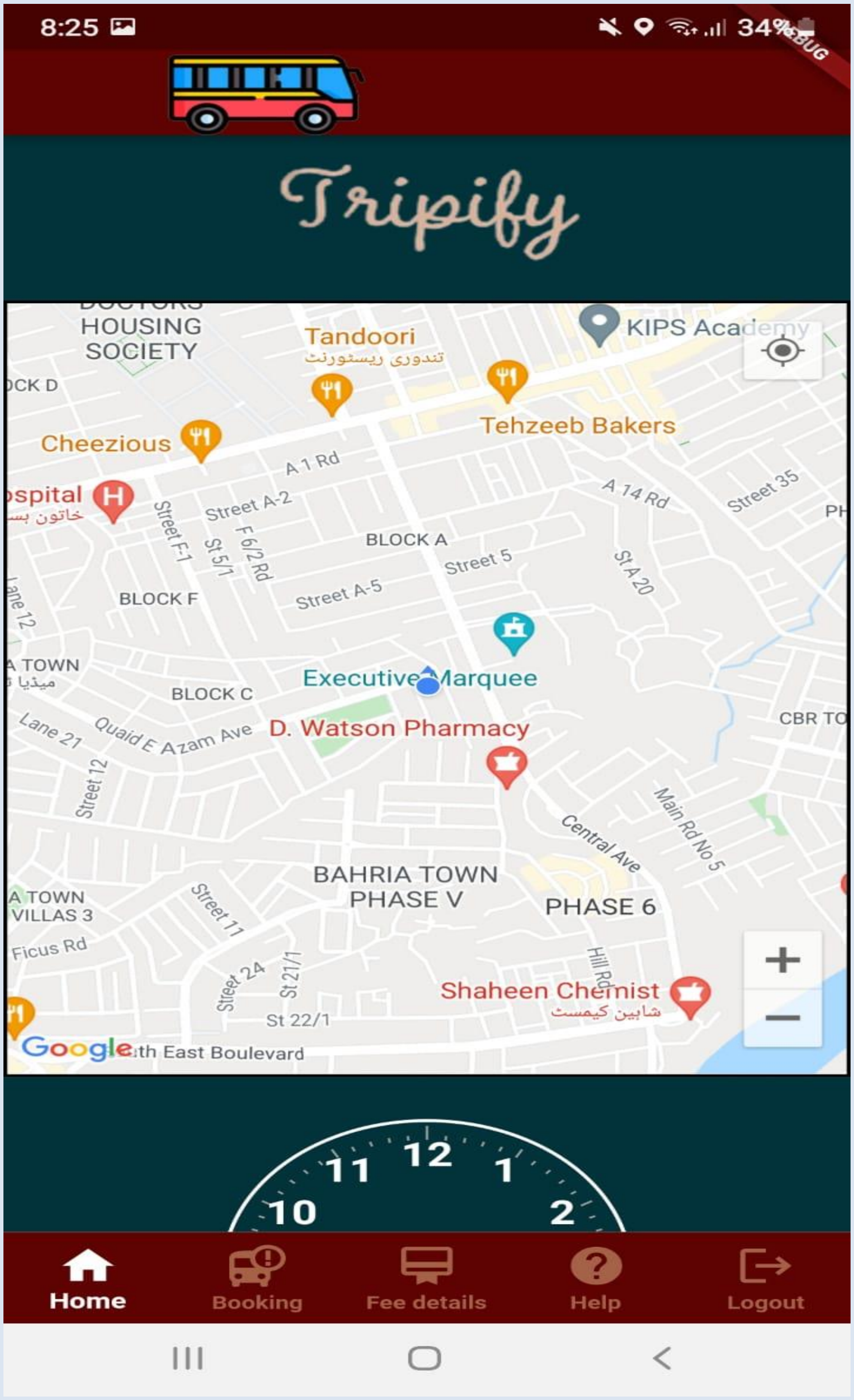
Username

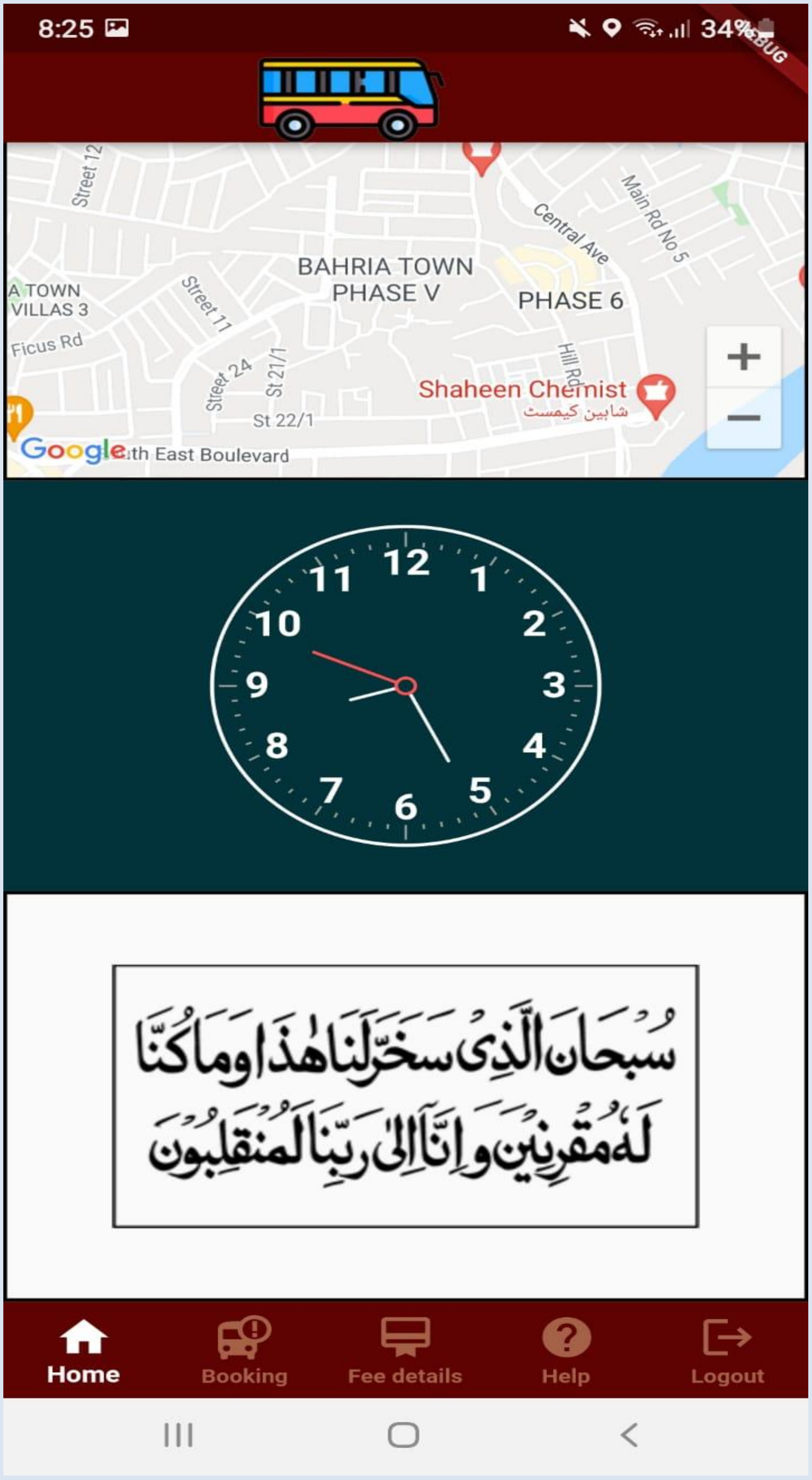
Password

Student

✓ By checking you agree to the Privacy Policies of Tripify

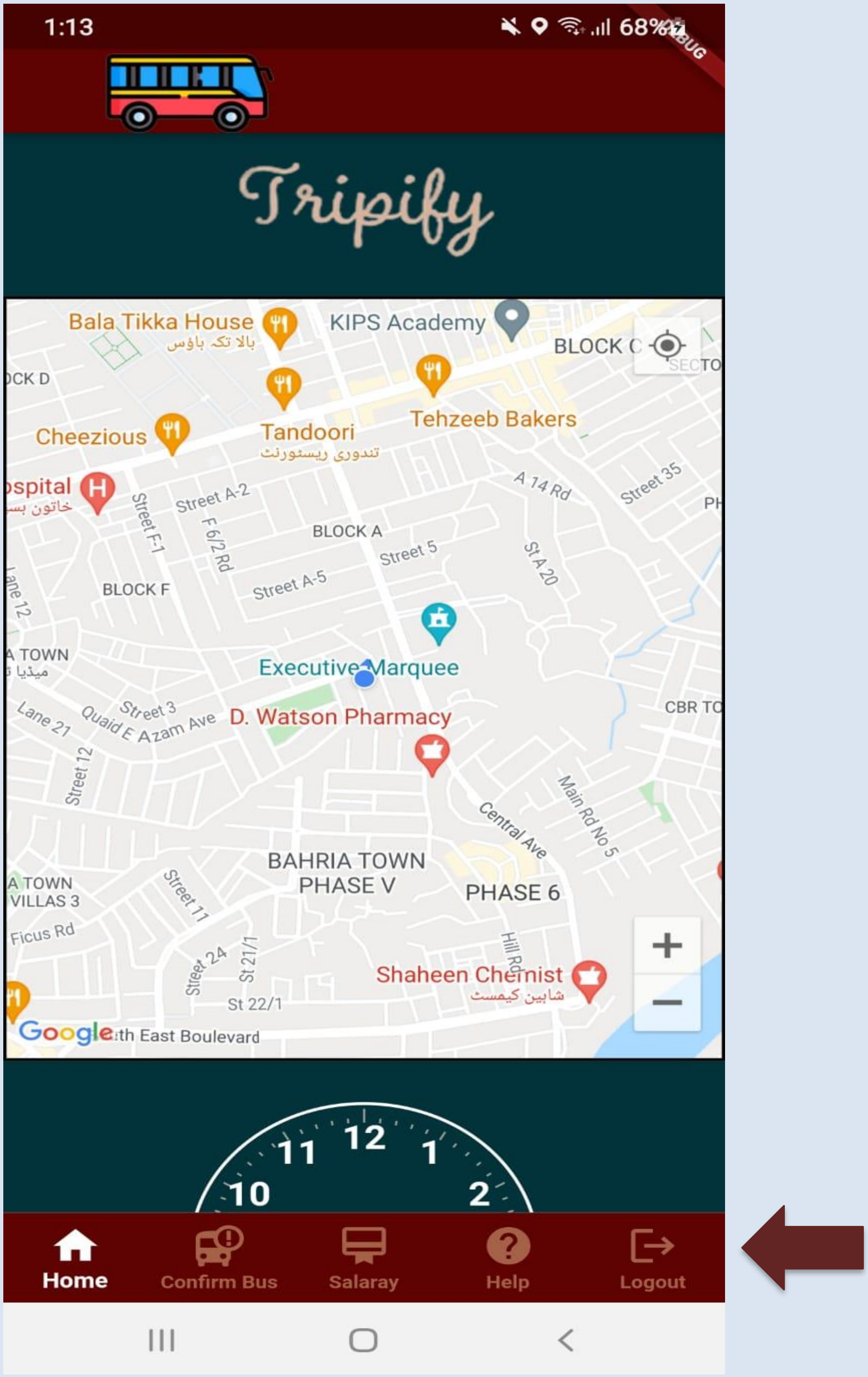
- **Students home page:**



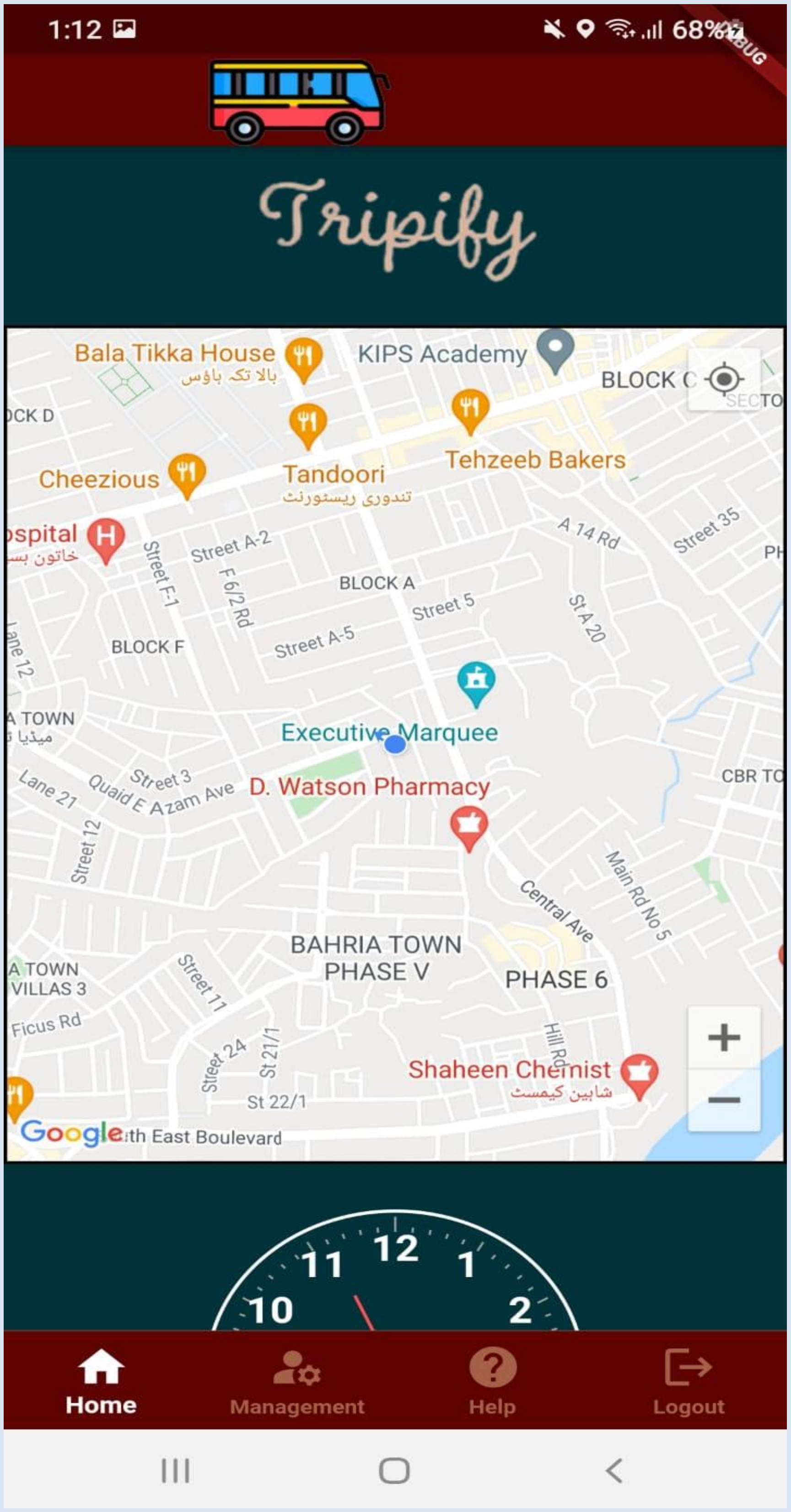


سُبحان الَّذِی سَخَّرَلَنَا هٰذَا وَمَا كُنَّا لَهٗ مُقْرِنِيْنَ وَاِنَّآ اِلٰى رَبِّنَا لَمُنْقَلِبُوْن

- **Drivers home page:**

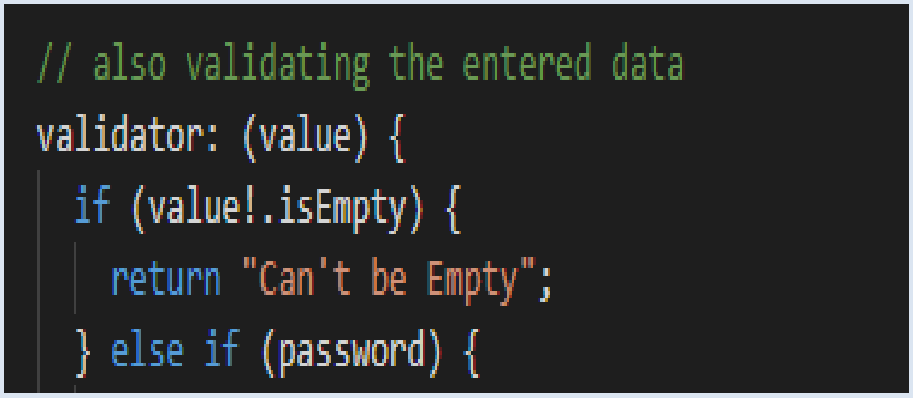


- **Admins home page:**

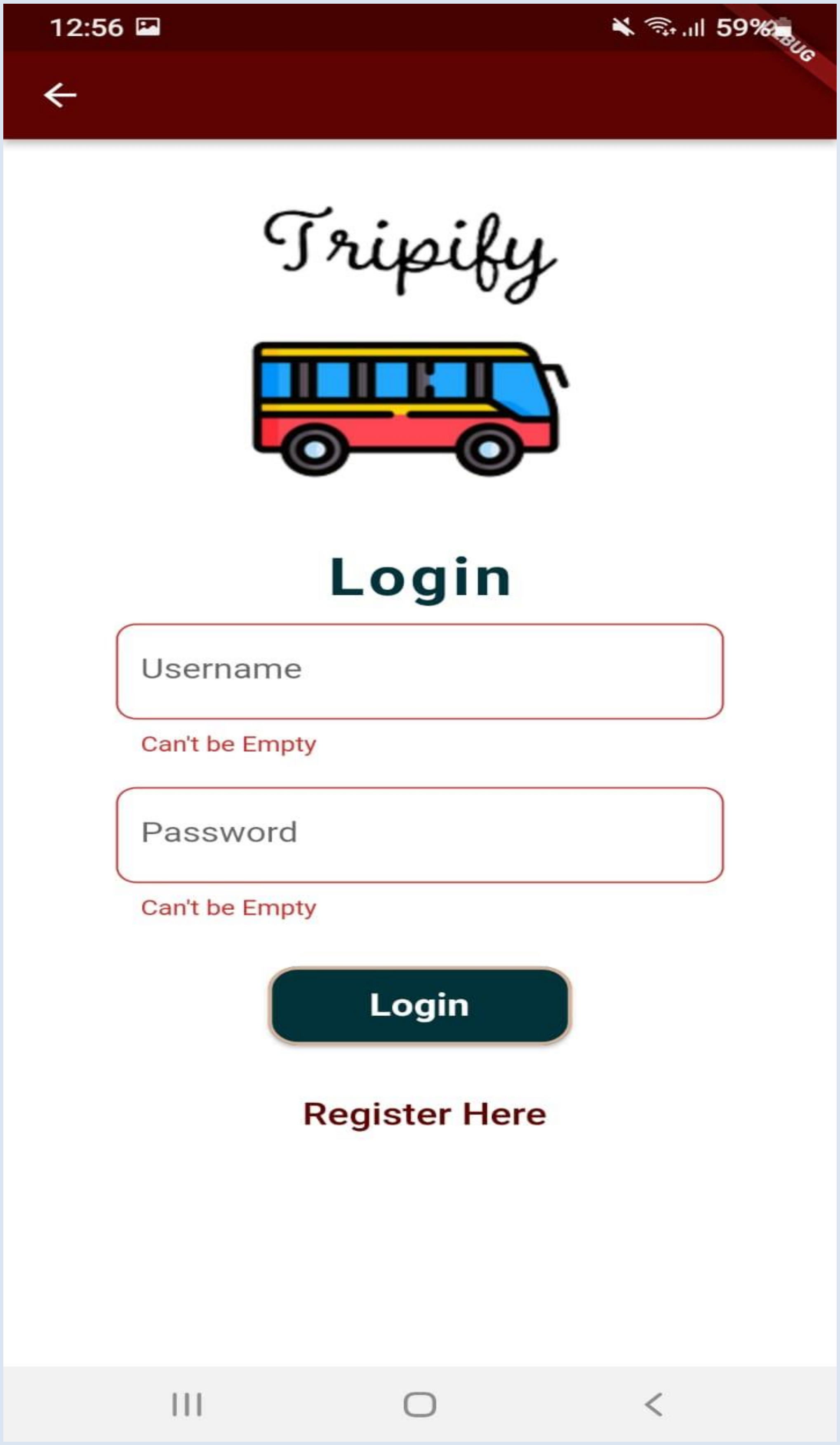


Checks/ possible outputs:

1. No field can be left empty.



```
// also validating the entered data
validator: (value) {
  if (value!.isEmpty) {
    return "Can't be Empty";
  } else if (password) {
```


12:56 59%

Tripify

Login

Username

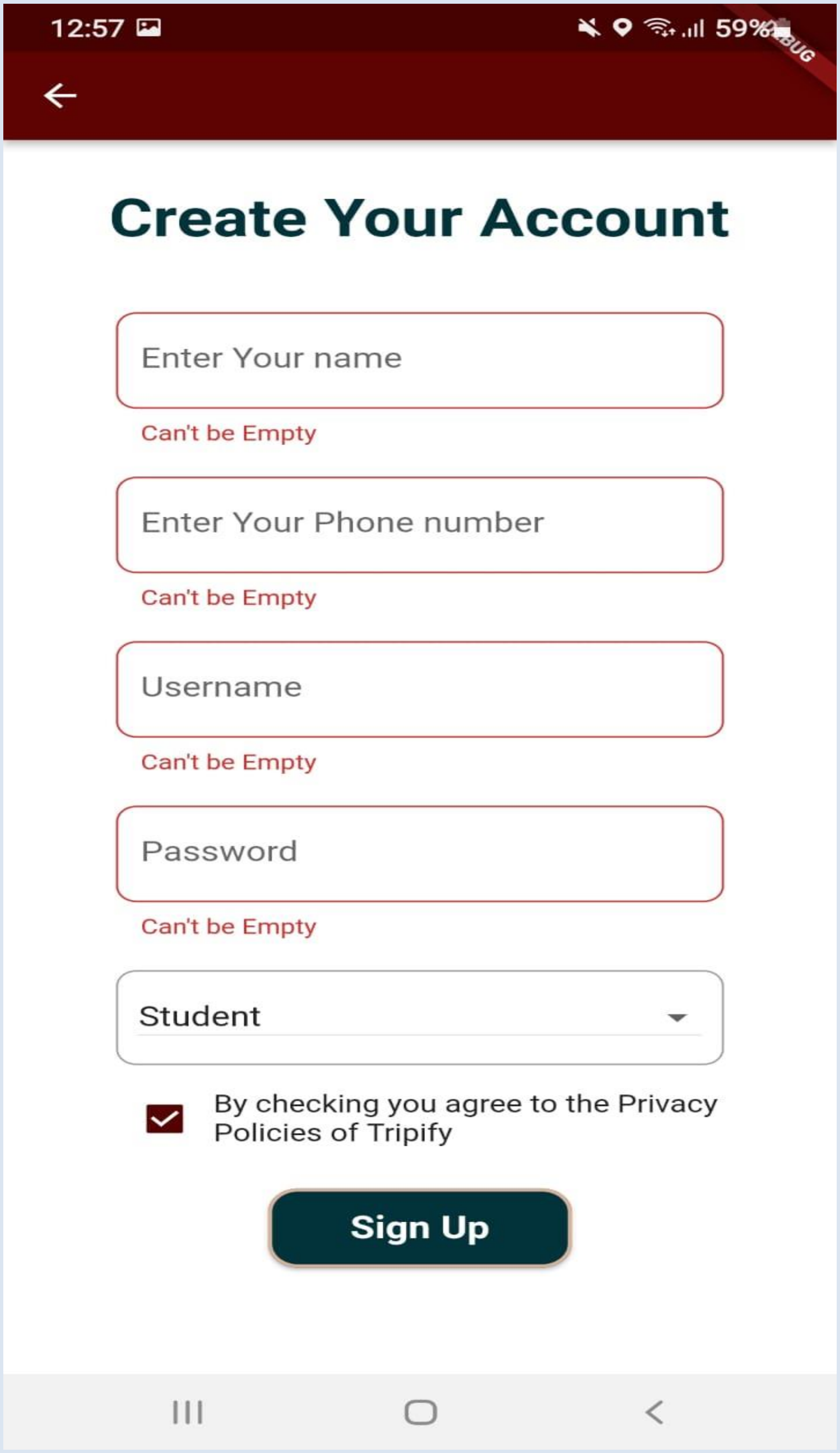
Can't be Empty

Password

Can't be Empty

Login

Register Here



Create Your Account

Enter Your name

Can't be Empty

Enter Your Phone number

Can't be Empty

Username

Can't be Empty

Password

Can't be Empty

Student

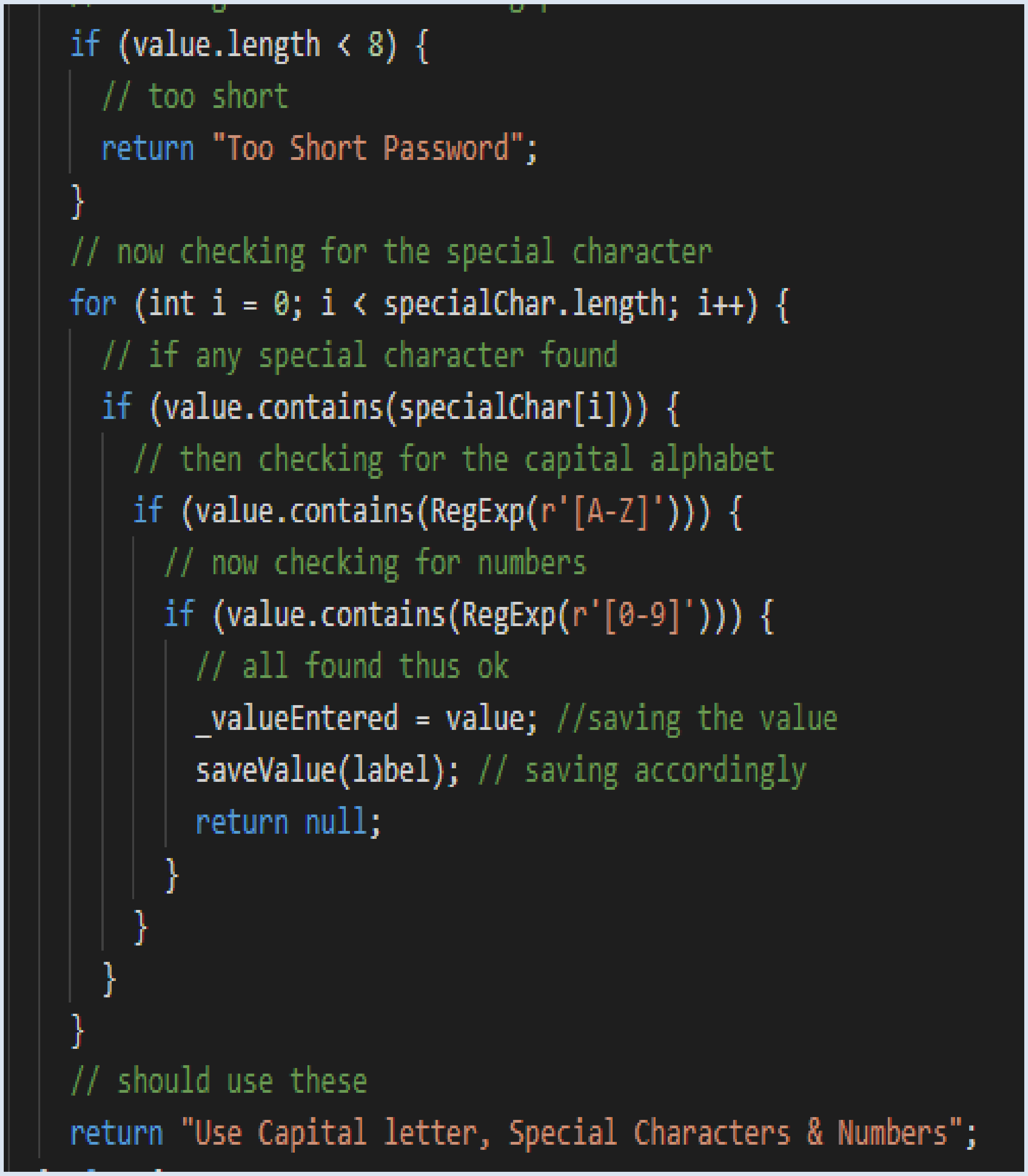
✓ By checking you agree to the Privacy Policies of Tripify

Sign Up

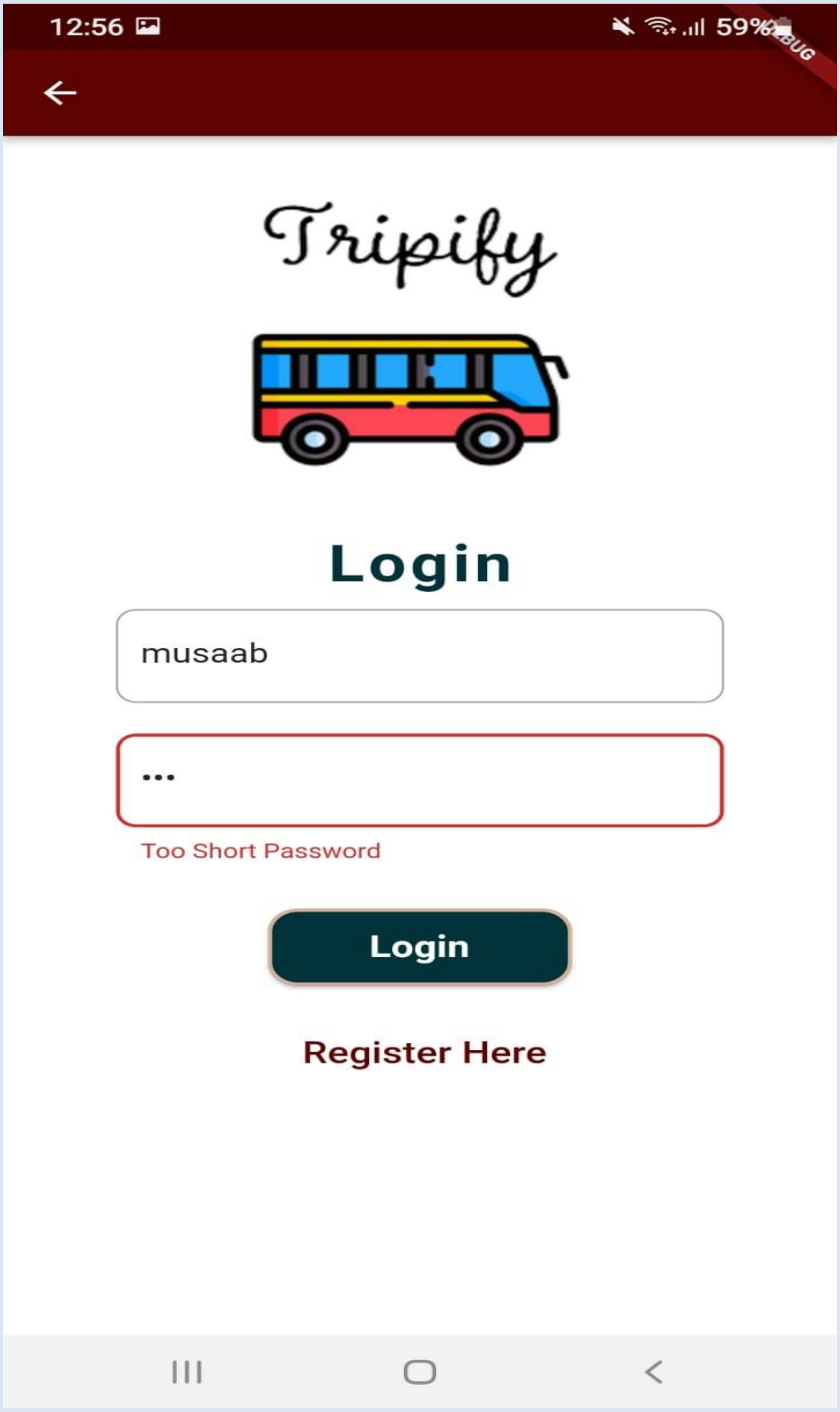
2. The password checks:

a. It can’t be less than 8 characters.

b. It should contain capital letters, small letters, numbers, special characters.



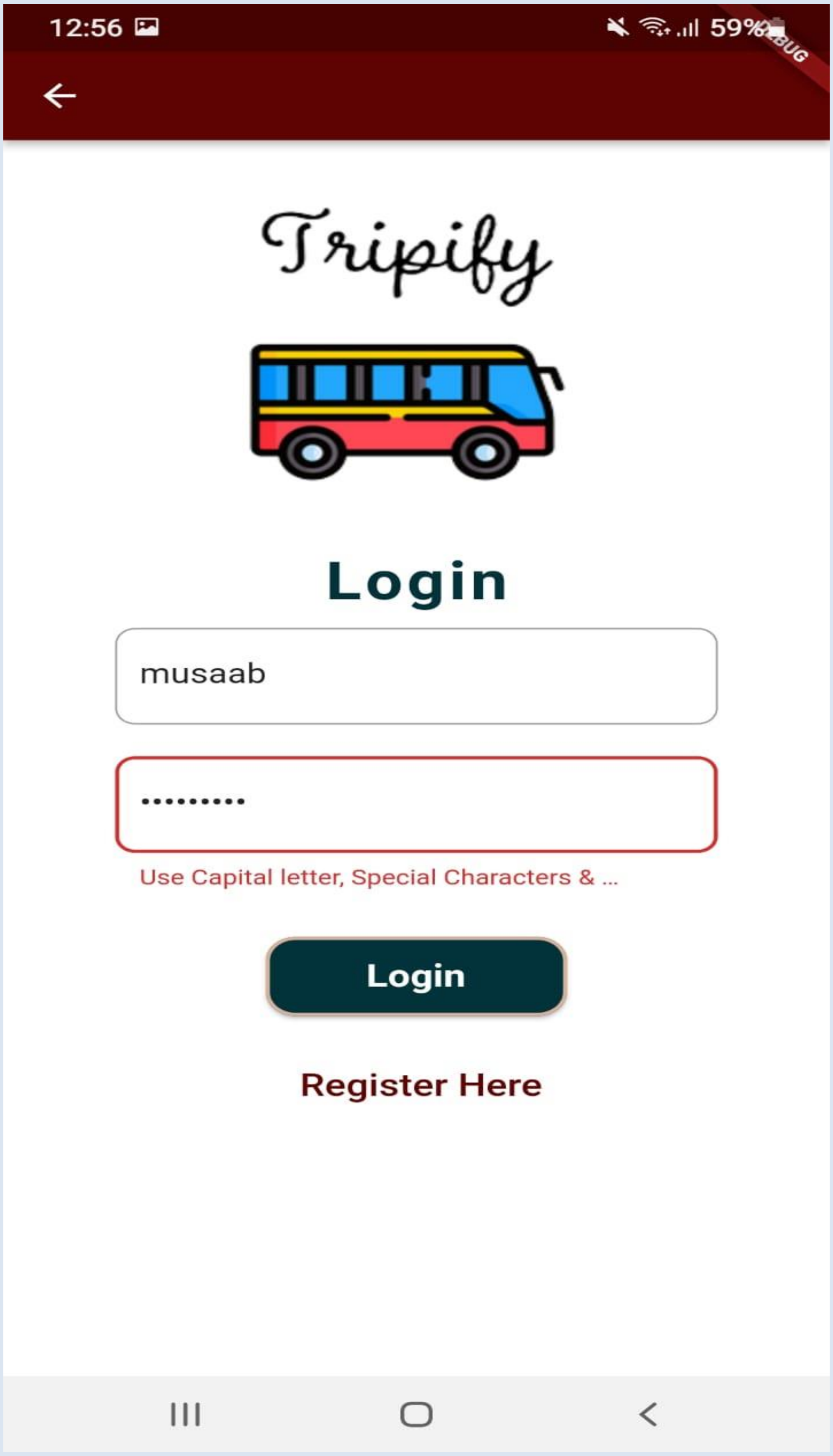
```
if (value.length < 8) {
  // too short
  return "Too Short Password";
}
// now checking for the special character
for (int i = 0; i < specialChar.length; i++) {
  // if any special character found
  if (value.contains(specialChar[i])) {
    // then checking for the capital alphabet
    if (value.contains(RegExp(r'[A-Z]'))) {
      // now checking for numbers
      if (value.contains(RegExp(r'[0-9]'))) {
        // all found thus ok
        _valueEntered = value; //saving the value
        saveValue(label); // saving accordingly
        return null;
      }
    }
  }
}
// should use these
return "Use Capital letter, Special Characters & Numbers";
```



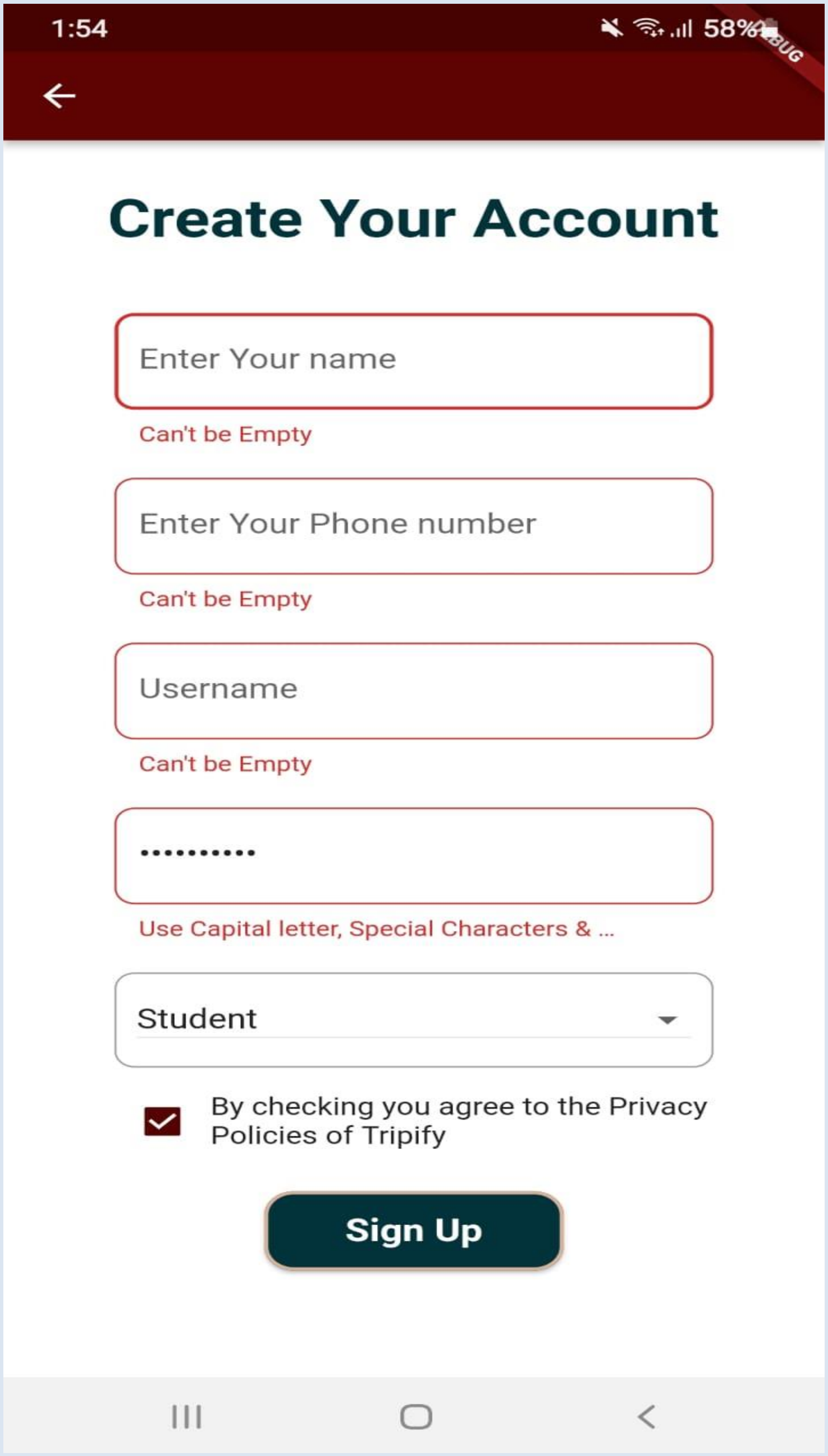
musaab

•••

Too Short Password


12:56
59%
DEBUG
Tripify
Login
musaab
••••••••••
Use Capital letter, Special Characters & ...
Login
Register Here

3.



1:54

58%

DEBUG

Create Your Account

Enter Your name

Can't be Empty

Enter Your Phone number

Can't be Empty

Username

Can't be Empty

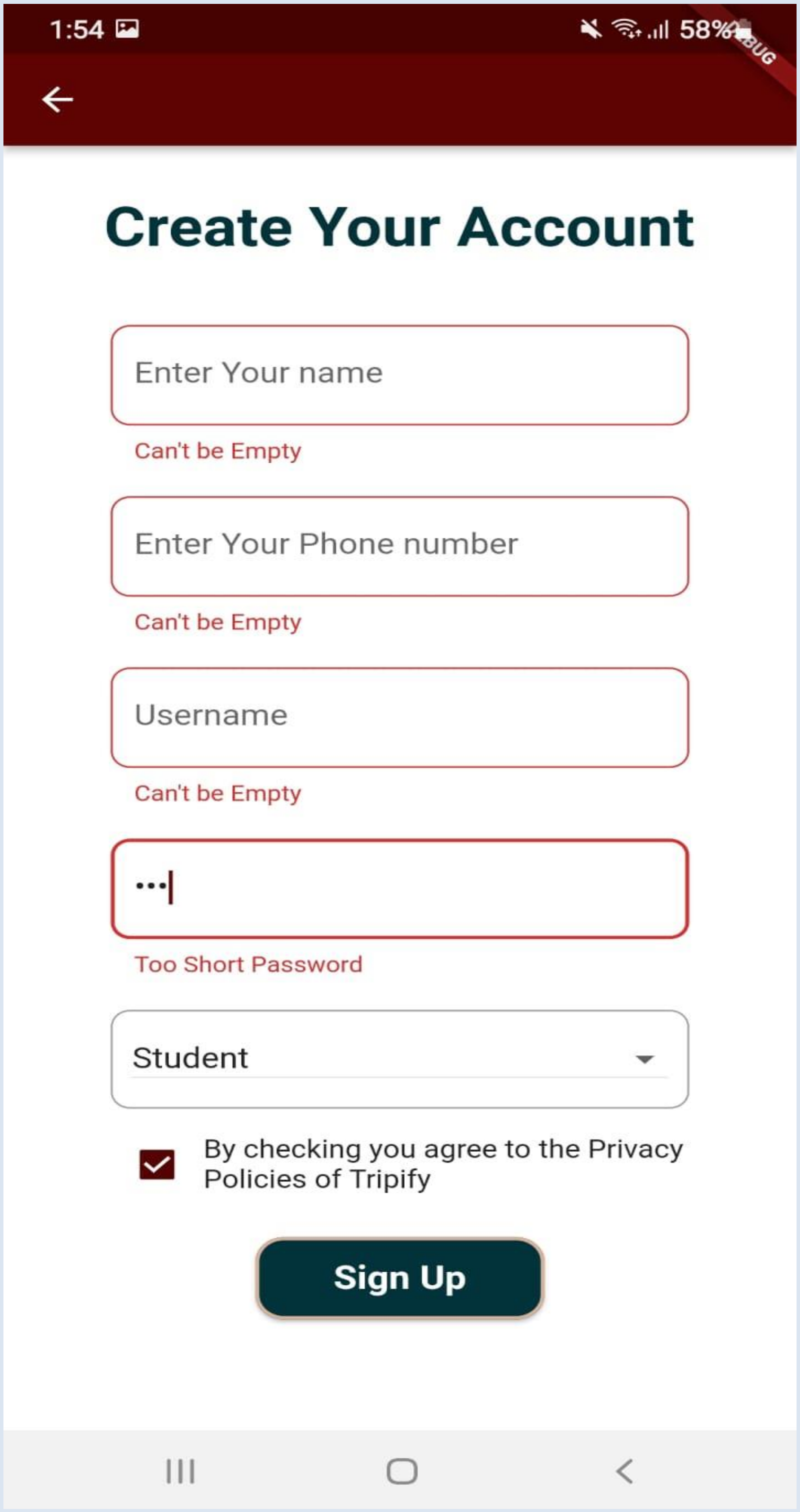
•••••••••••

Use Capital letter, Special Characters & ...

Student

✓ By checking you agree to the Privacy Policies of Tripify

Sign Up


1:54 58%

Create Your Account

Enter Your name

Can't be Empty

Enter Your Phone number

Can't be Empty

Username

Can't be Empty

•••

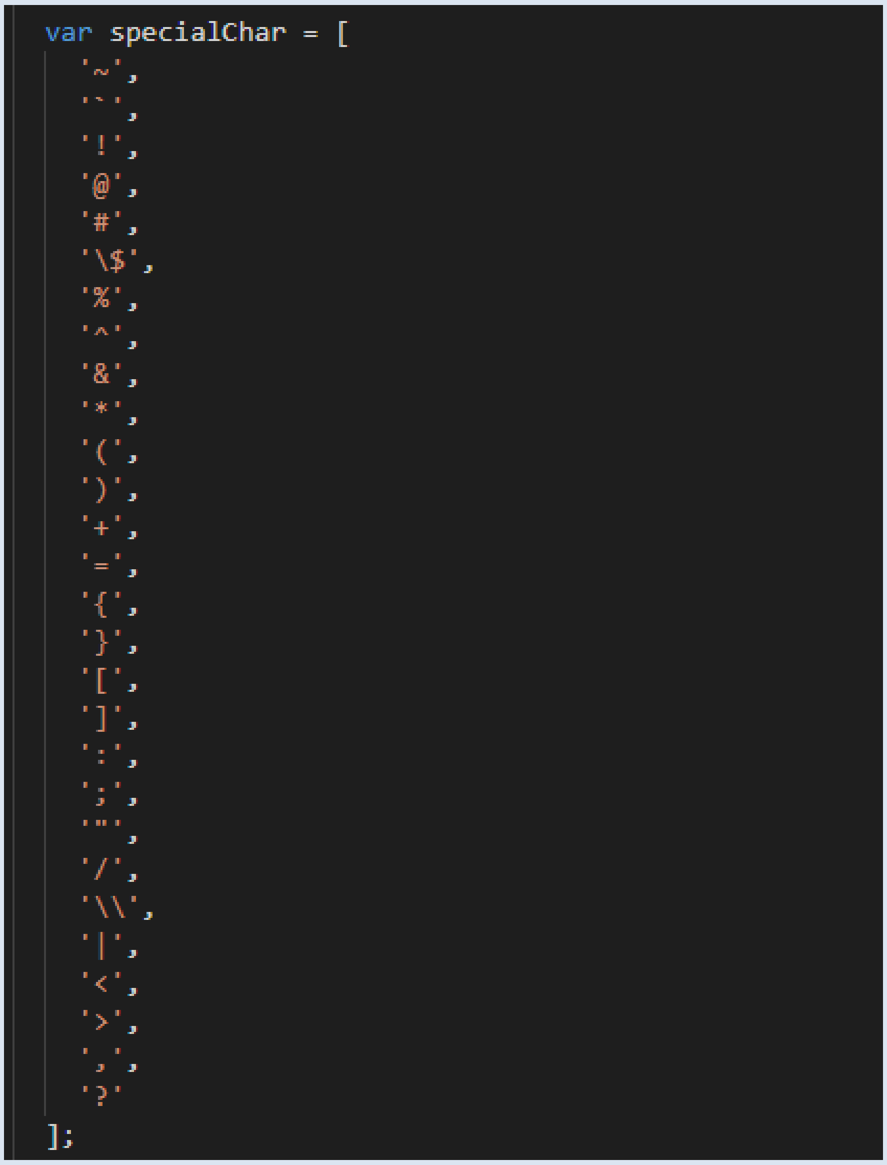
Too Short Password

Student

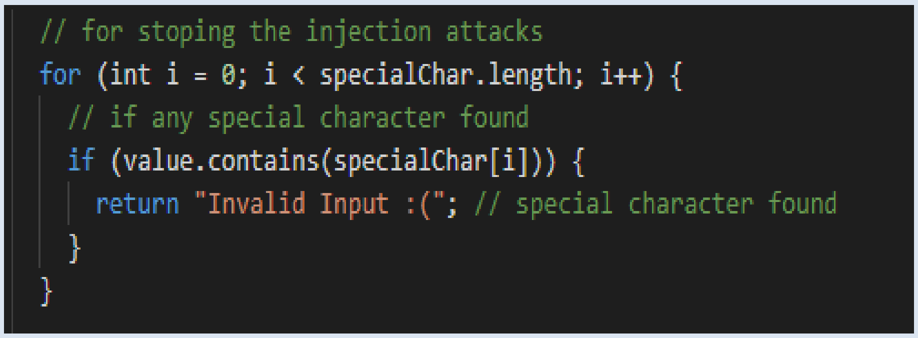
By checking you agree to the Privacy Policies of Tripify

Sign Up

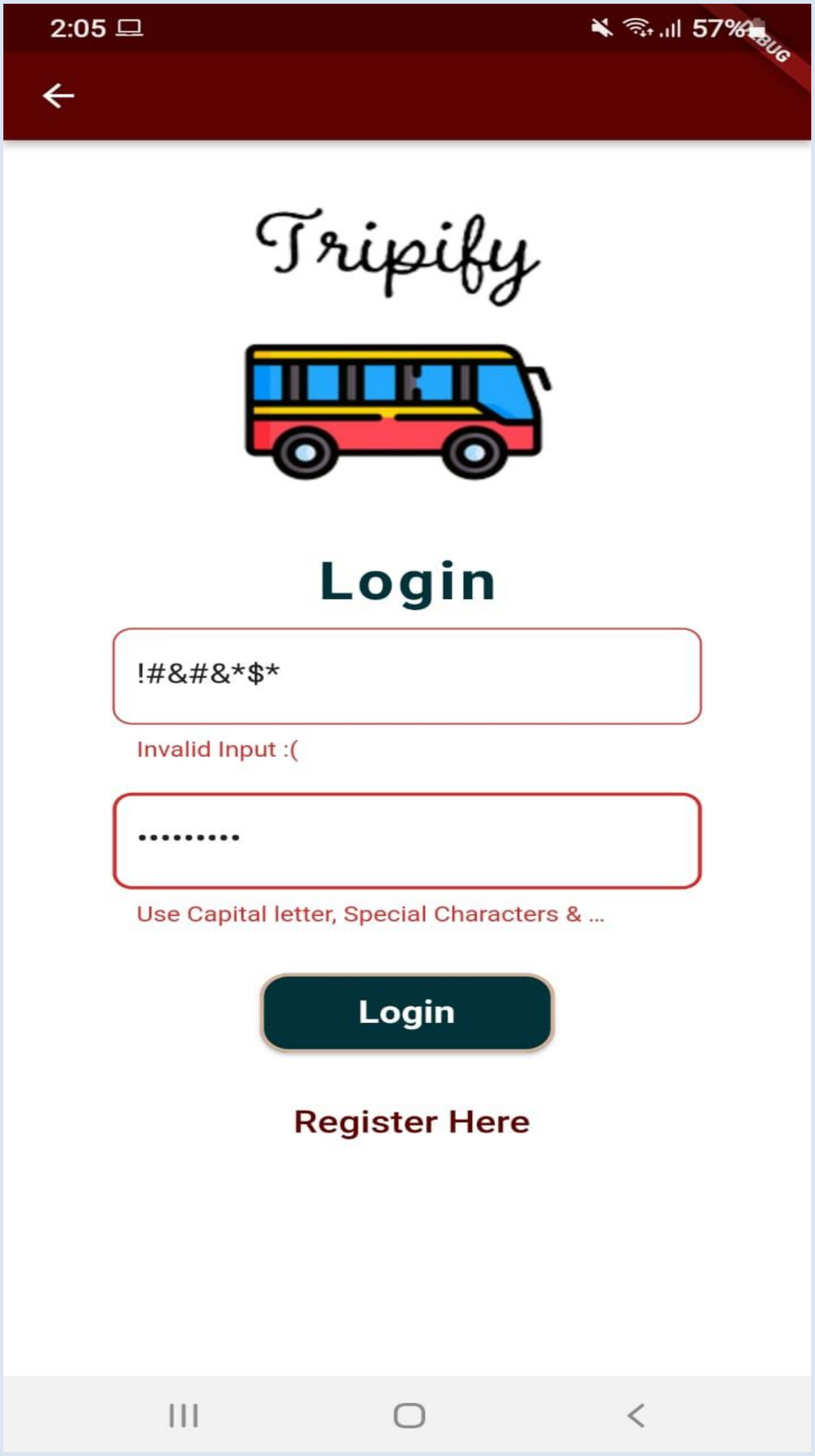
3. Check against injection attacks.



```
var specialChar = [
    '~',
    '`',
    '!',
    '@',
    '#',
    '\$',
    '%',
    '^',
    '&',
    '*',
    '(',
    ')',
    '+',
    '=',
    '{',
    '}',
    '[',
    ']',
    ':',
    ';',
    '"',
    '/',
    '\\',
    '|',
    '<',
    '>',
    ',',
    '?'
];
```



```
// for stoping the injection attacks
for (int i = 0; i < specialChar.length; i++) {
  // if any special character found
  if (value.contains(specialChar[i])) {
    return "Invalid Input :("; // special character found
  }
}
```



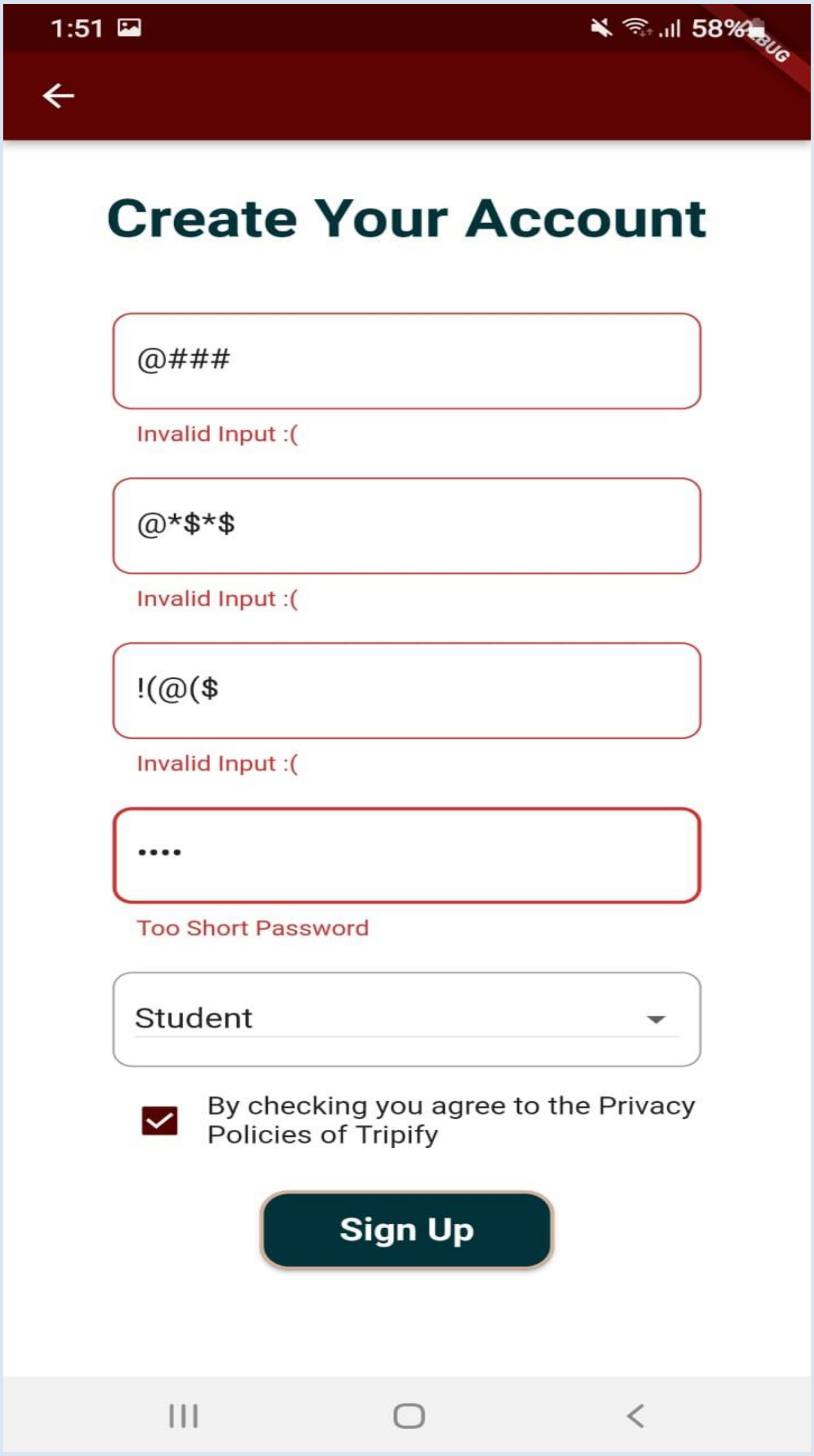
Login

!#&#&*$*

Invalid Input :(

••••••••••

Use Capital letter, Special Characters & ...


1:51

58%

Create Your Account

@###

Invalid Input :(

@*$*$

Invalid Input :(

!(@($

Invalid Input :(

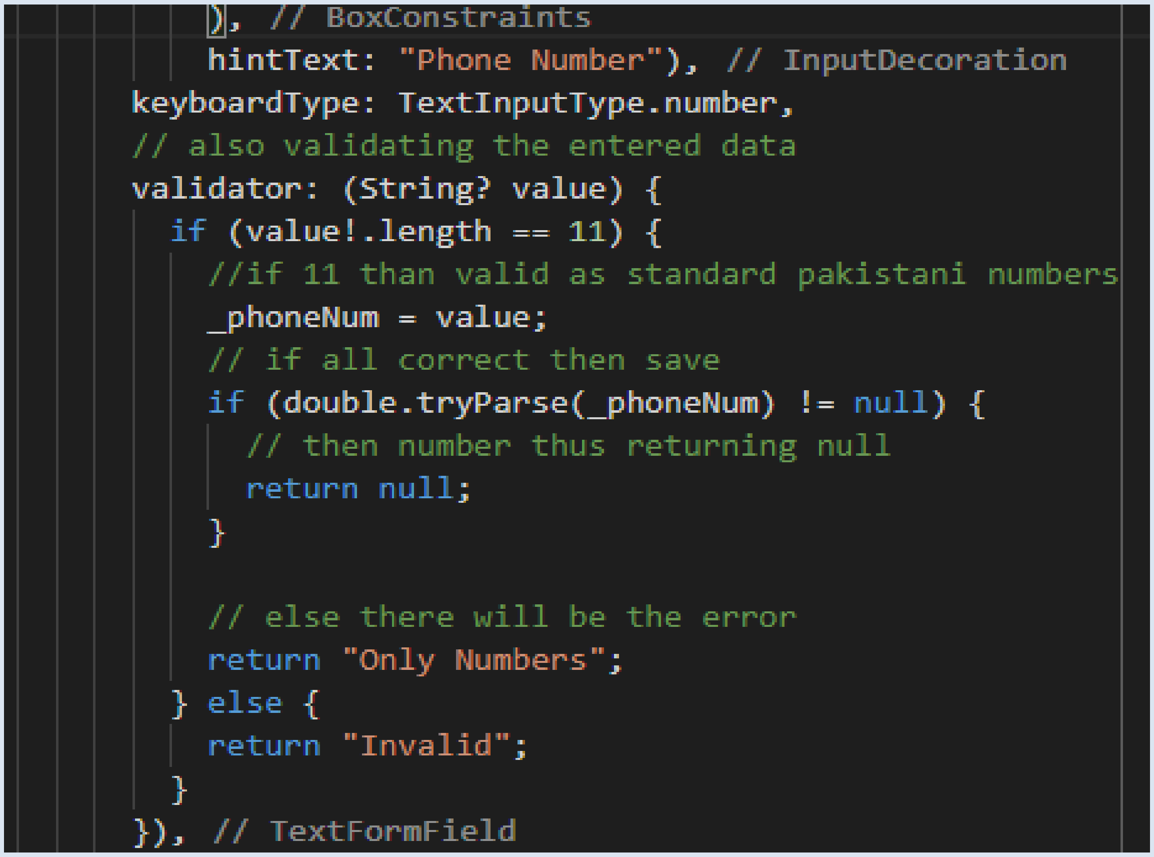
••••

Too Short Password

Student

✓ By checking you agree to the Privacy Policies of Tripify

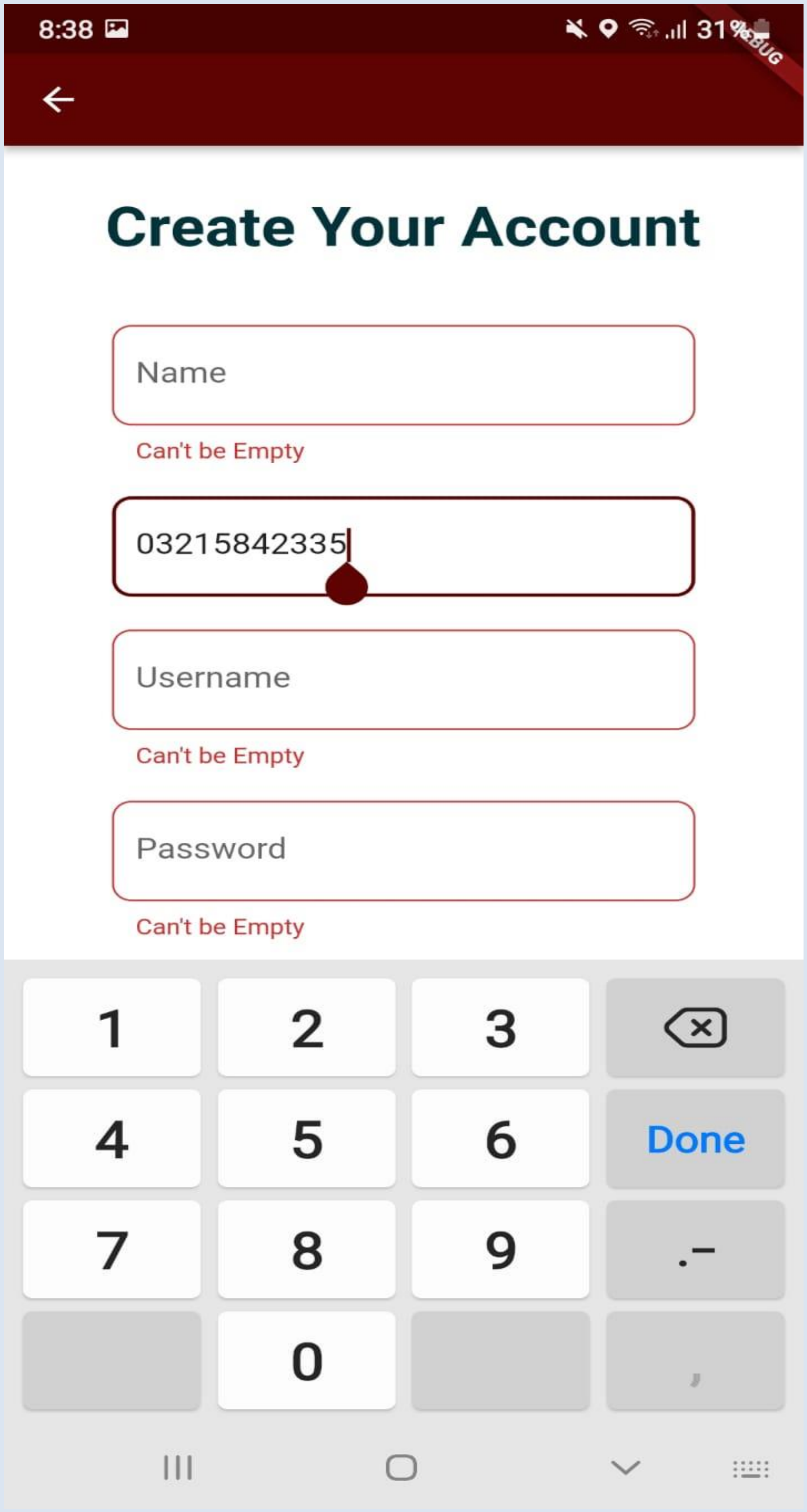
4. Check the phone number.



```
            ), // BoxConstraints
            hintText: "Phone Number"), // InputDecoration
        keyboardType: TextInputType.number,
        // also validating the entered data
        validator: (String? value) {
          if (value!.length == 11) {
            //if 11 than valid as standard pakistani numbers
            _phoneNum = value;
            // if all correct then save
            if (double.tryParse(_phoneNum) != null) {
              // then number thus returning null
              return null;
            }

            // else there will be the error
            return "Only Numbers";
          } else {
            return "Invalid";
          }
        }), // TextFormField
```

By this code snippet, we can see the respective output and the number can’t be more than 11 digits, and t we are not allowing it to enter any character other than numbers.



8:38

31%

DEBUG

Create Your Account

Name

Can't be Empty

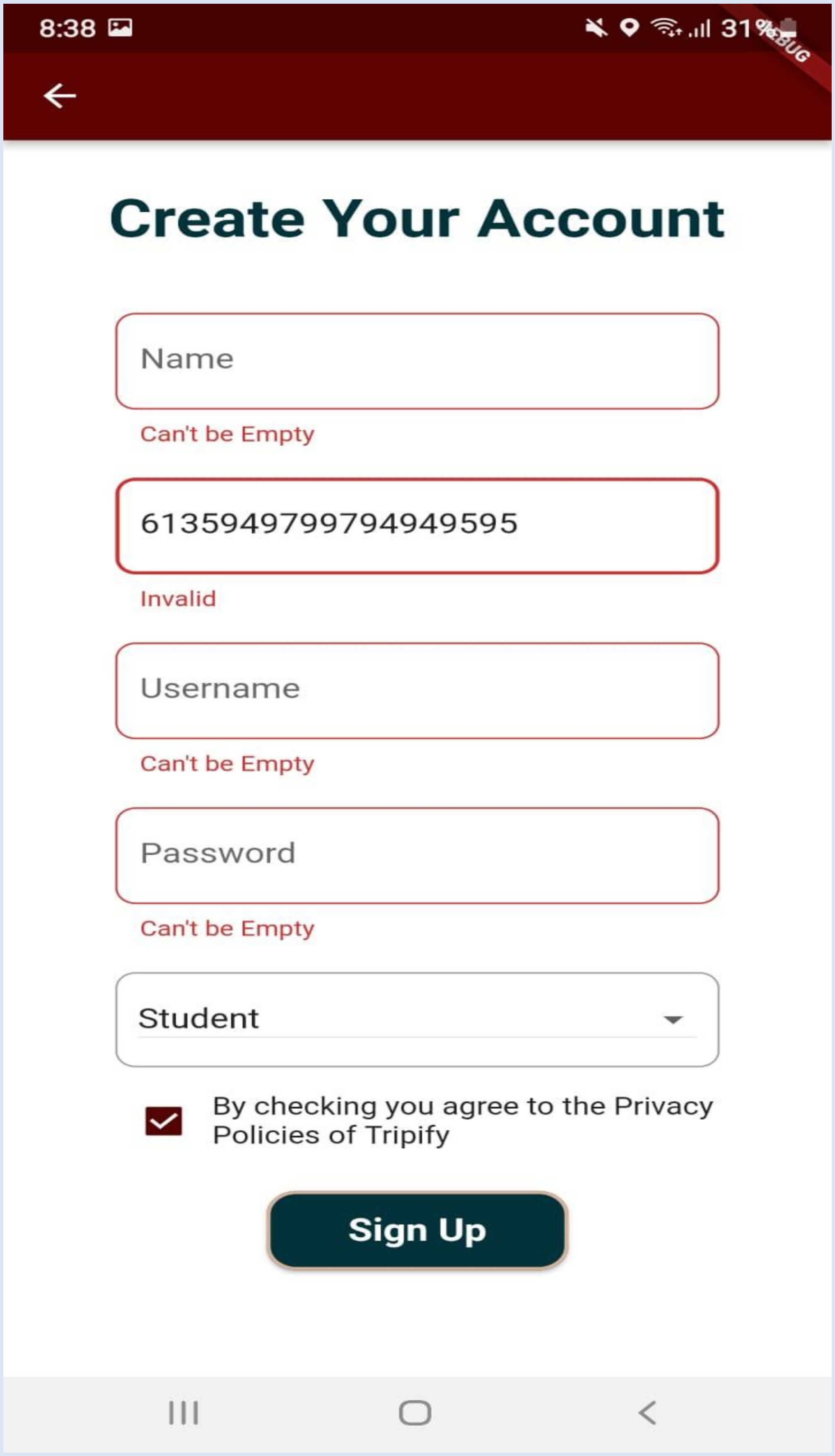
03215842335

Username

Can't be Empty

Password

Can't be Empty


8:38 31%

Create Your Account

Name

Can't be Empty

6135949799794949595

Invalid

Username

Can't be Empty

Password

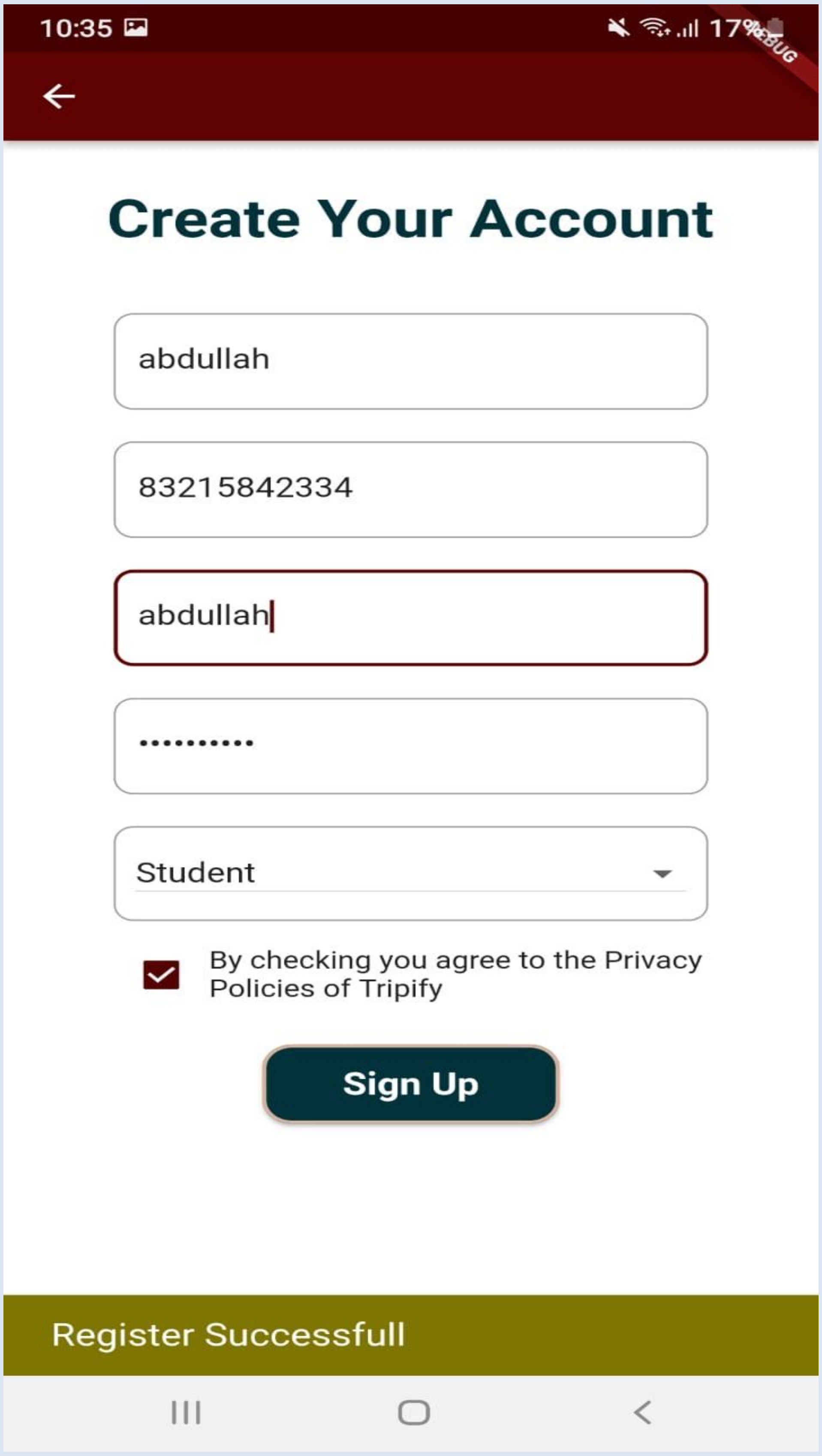
Can't be Empty

Student

[✓] By checking you agree to the Privacy Policies of Tripify

Back-end Implementation: Main Functionalities

- **Creating the User Account**



10:35

17%

DEBUG

Create Your Account

abdullah

83215842334

abdullah

••••••••••

Student

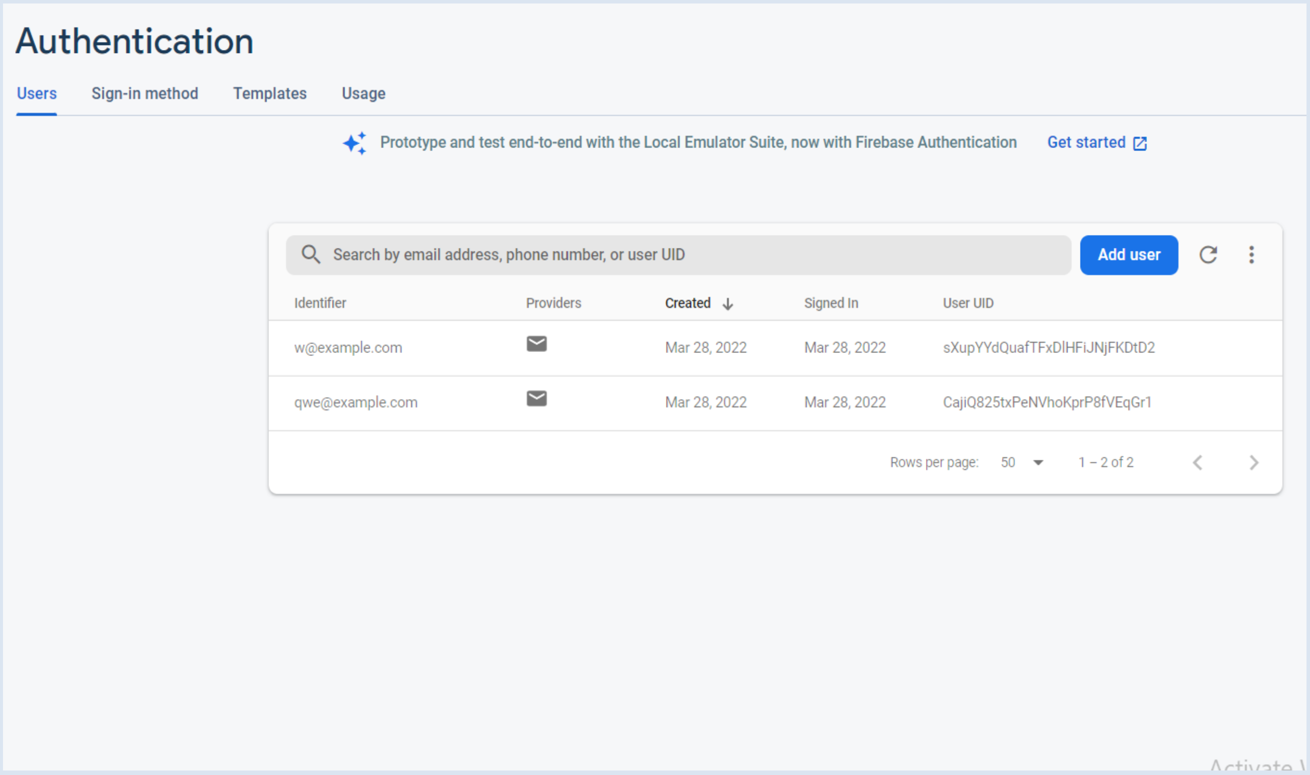
✓ By checking you agree to the Privacy Policies of Tripify

Sign Up

Register Successfull

- **Register Users**

Firebase will be handling the authentication for the users. We will be using Firebase flutter Function to add functionality to the application.



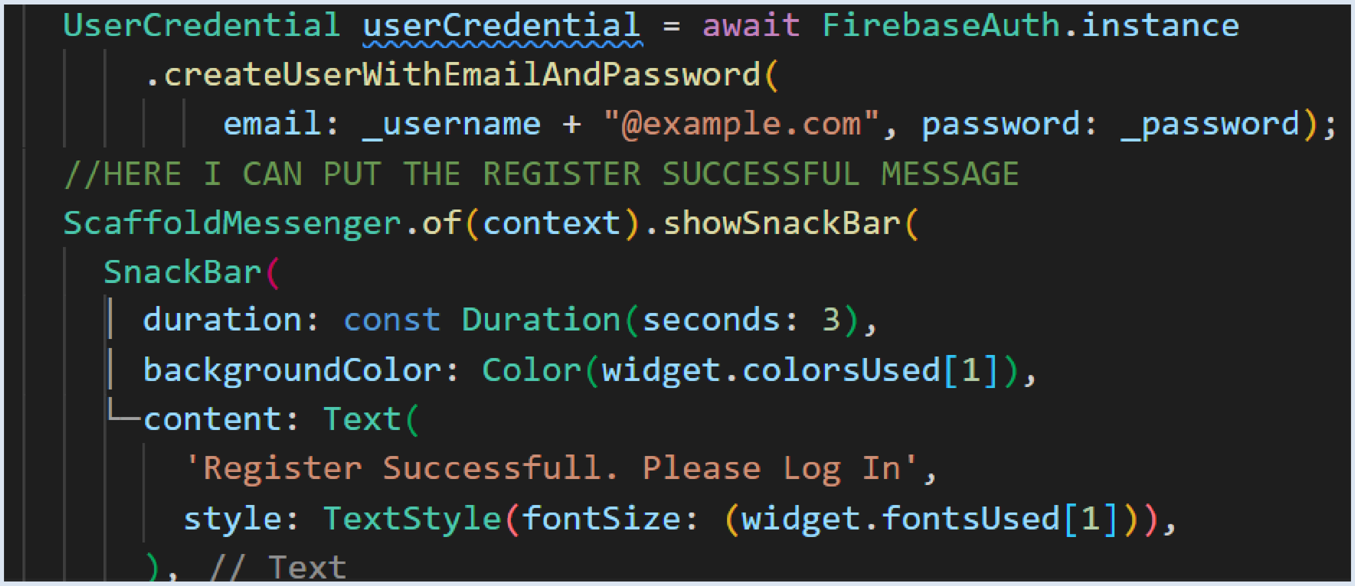
Authentication

Users
Sign-in method
Templates
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Search by email address, phone number, or user UID
Add user

Identifier	Providers	Created	Signed In	User UID
w@example.com	✉	Mar 28, 2022	Mar 28, 2022	sXupYYdQuafTFxDlHFiJNjFKDtD2
qwe@example.com	✉	Mar 28, 2022	Mar 28, 2022	CajiQ825txPeNVhoKprP8fVEqGr1

Rows per page: 50 1 – 2 of 2

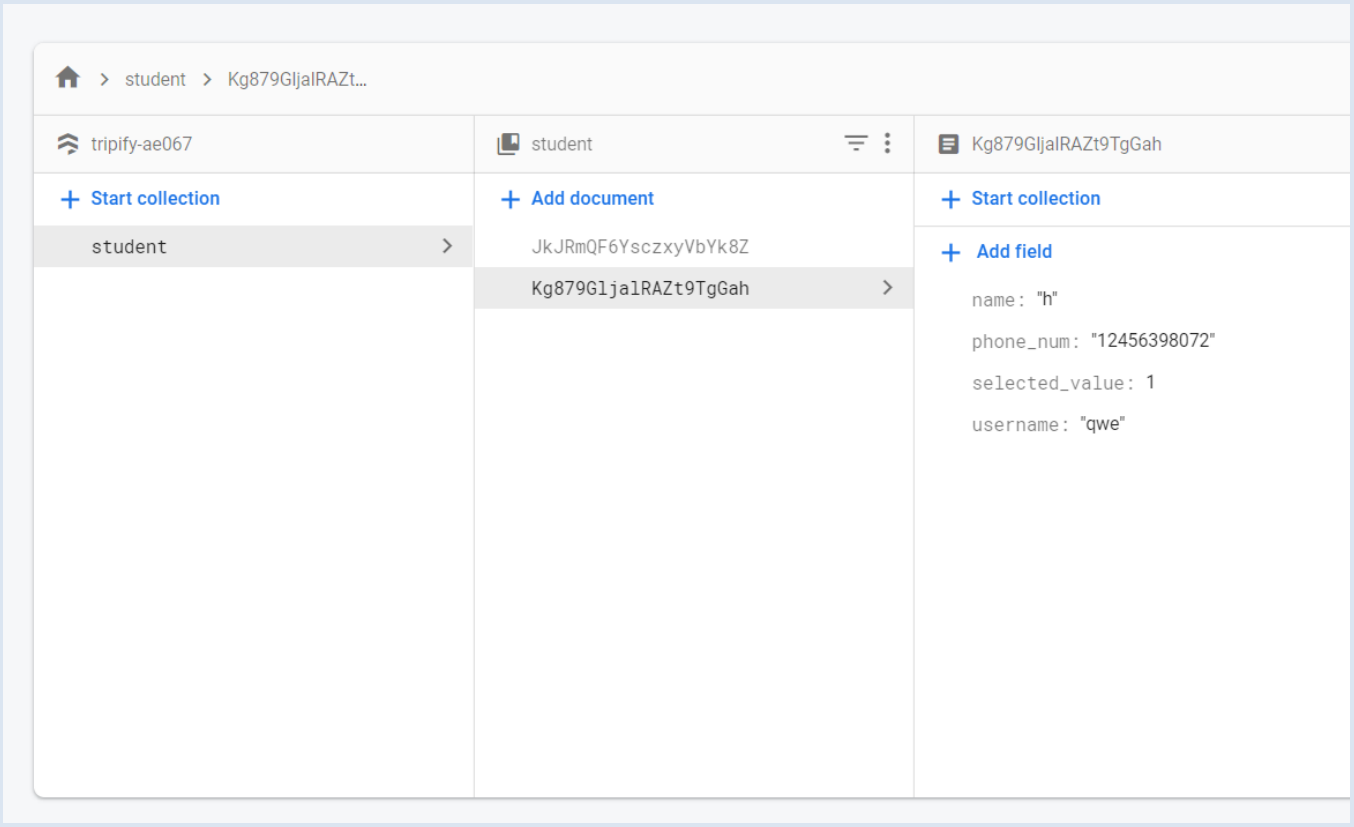
A code snippet sample that adds the User to the Firebase and Handles its authentication.



```
UserCredential userCredential = await FirebaseAuth.instance
    .createUserWithEmailAndPassword(
      email: _username + "@example.com", password: _password);
//HERE I CAN PUT THE REGISTER SUCCESSFUL MESSAGE
ScaffoldMessenger.of(context).showSnackBar(
  SnackBar(
    duration: const Duration(seconds: 3),
    backgroundColor: Color(widget.colorsUsed[1]),
    content: Text(
      'Register Successfull. Please Log In',
      style: TextStyle(fontSize: (widget.fontsUsed[1])),
    ), // Text
```

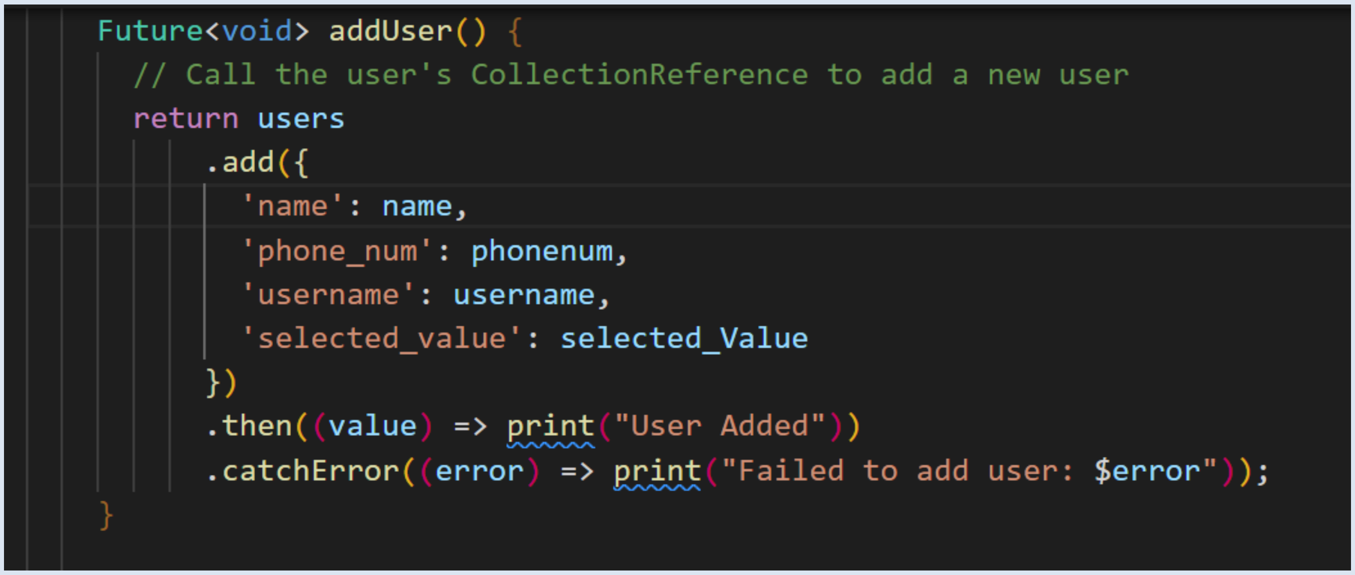
- ## Saving the Data to Database

This is the Fire store database that is being used by the developers for this project.

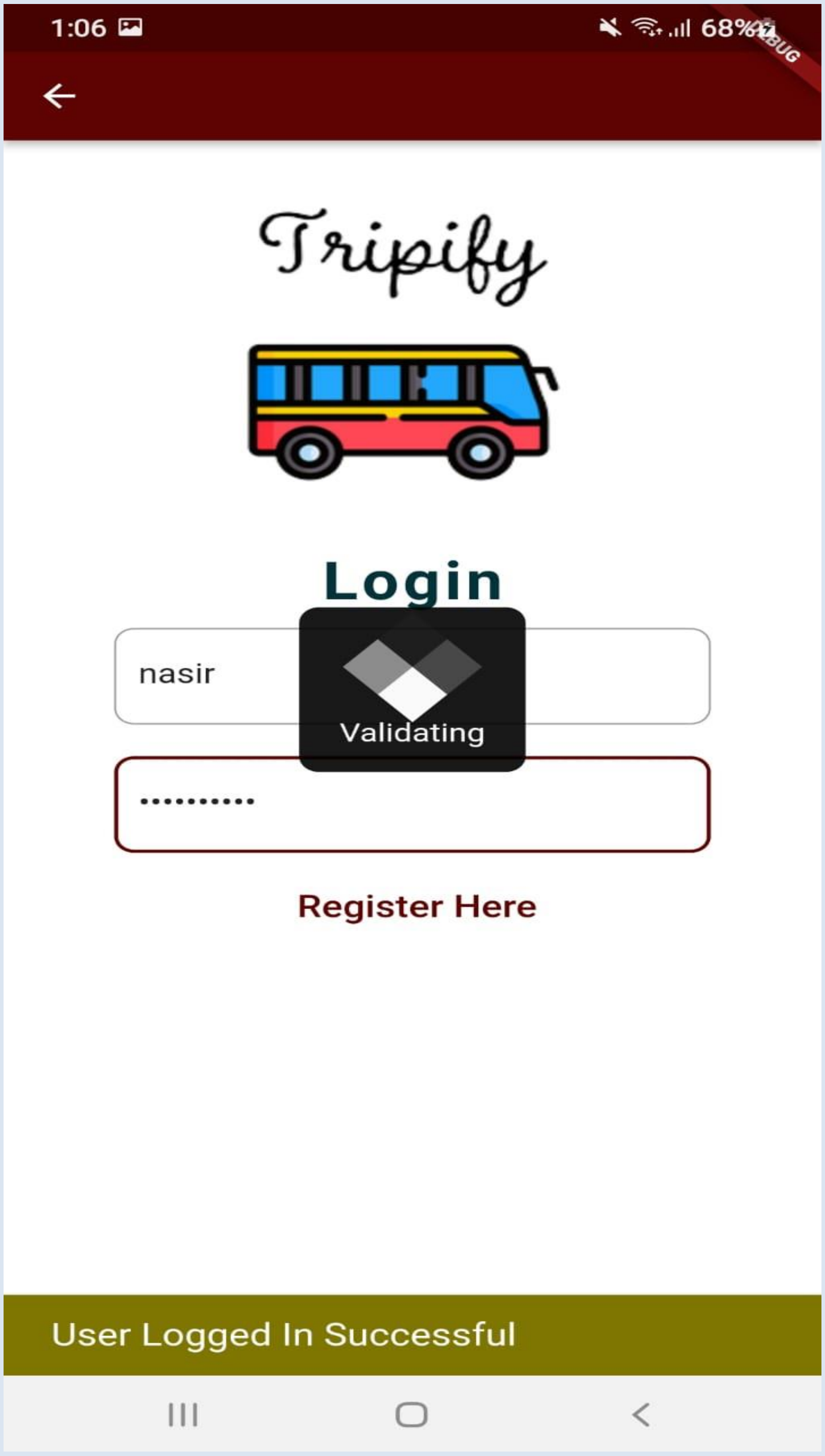


```
name: "h"
phone_num: "12456398072"
selected_value: 1
username: "qwe"
```

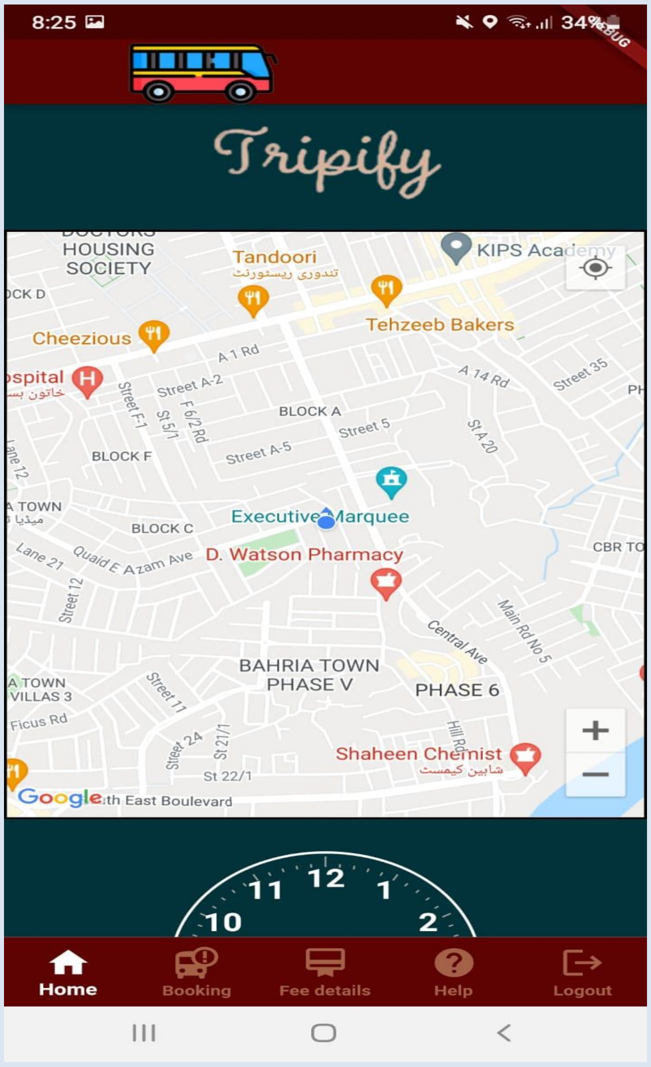
The information below will be saved in the database when the user will first registers in the application.



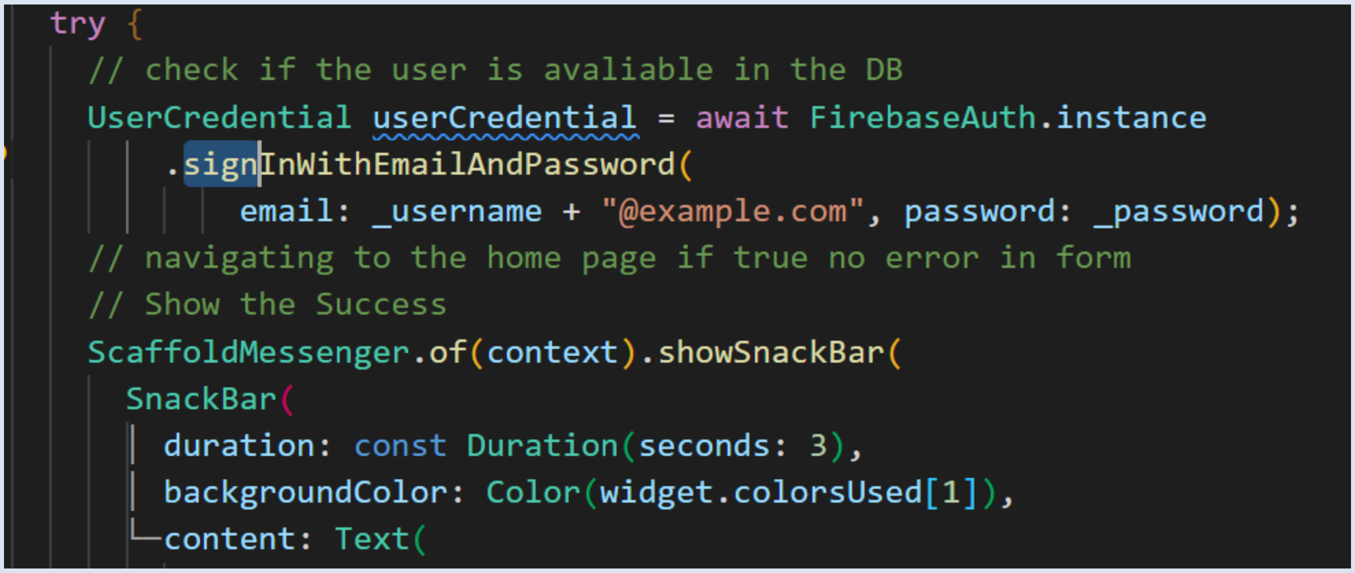
```
Future<void> addUser() {
  // Call the user's CollectionReference to add a new user
  return users
      .add({
        'name': name,
        'phone_num': phonenum,
        'username': username,
        'selected_value': selected_Value
      })
      .then((value) => print("User Added"))
      .catchError((error) => print("Failed to add user: $error"));
}
```



User Logged In Successful



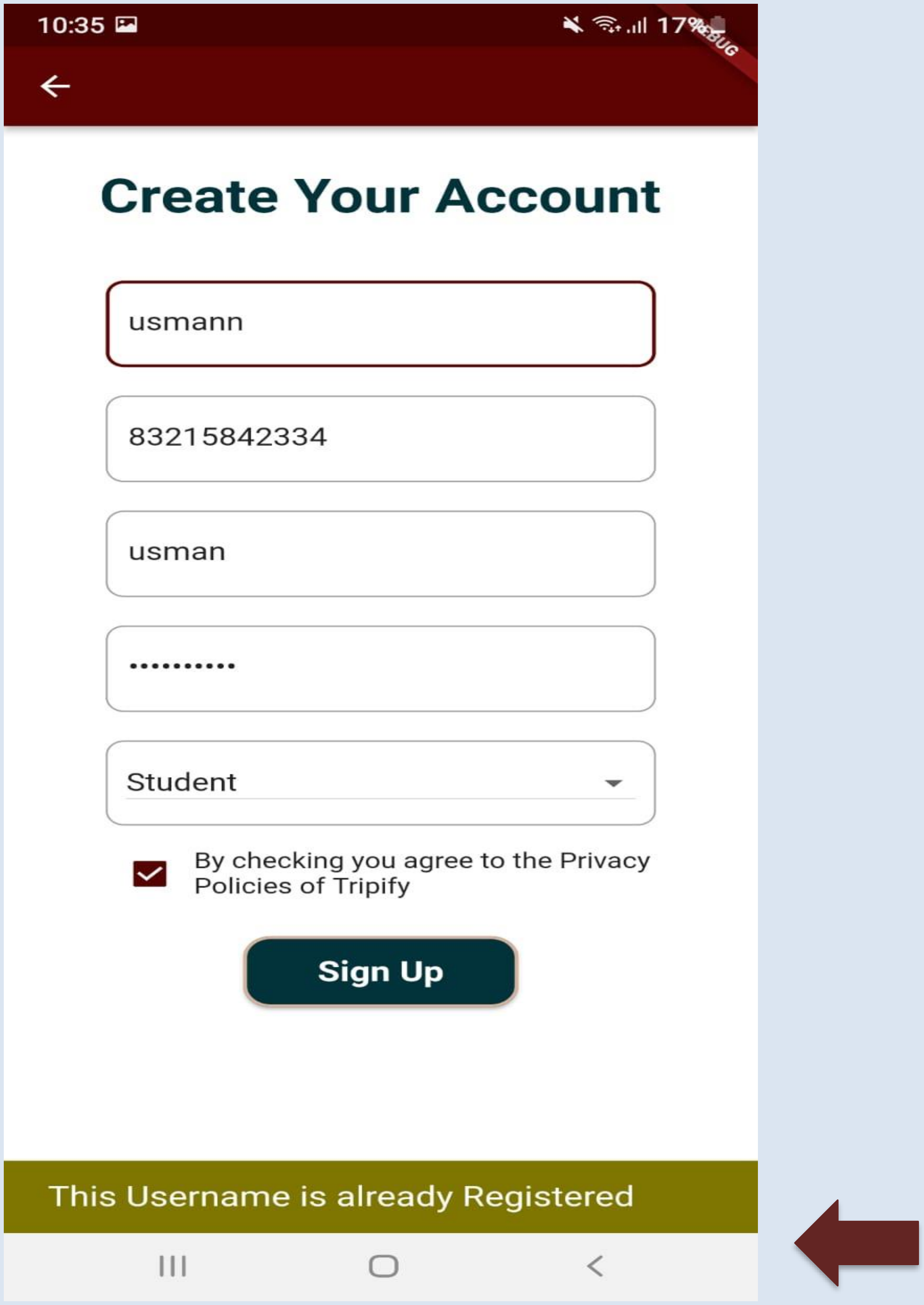
After Successfully validating from the server the user is redirected to home whether it is a student, admin, or the driver.



```
try {
  // check if the user is avaliable in the DB
  UserCredential userCredential = await FirebaseAuth.instance
        .signInWithEmailAndPassword(
            email: _username + "@example.com", password: _password);
  // navigating to the home page if true no error in form
  // Show the Success
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      duration: const Duration(seconds: 3),
      backgroundColor: Color(widget.colorsUsed[1]),
      content: Text(
```

- **Server Validation for Secure Development**

The user account is validated correctly before they are created and we tried to make the application much more secure.



10:35

Create Your Account

usmann

83215842334

usman

•••••••••••

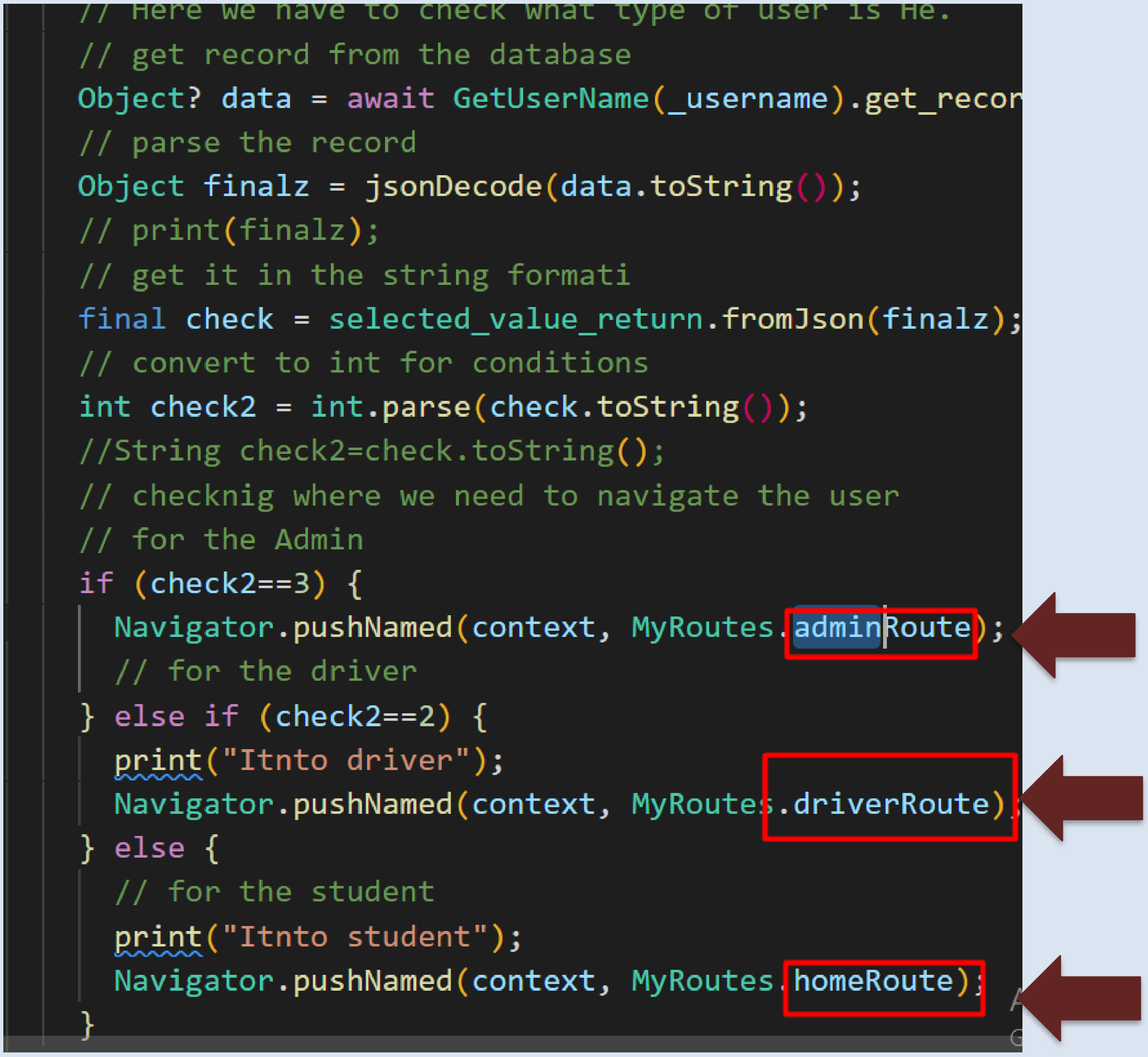
Student

✓ By checking you agree to the Privacy Policies of Tripify

This Username is already Registered

- **Routing to the Correct Home Page**

There are 3 user home pages in the application. We made sure when the user login to the application the data we checked from the database from which he created the account accordingly is routed to that particular page. The restriction that one user can never access the other homepage like a student can never access the driver homepage will be added in the account managing sprint.



```
// Here we have to check what type of user is He.
// get record from the database
Object? data = await GetUserName(_username).get_recor
// parse the record
Object finalz = jsonDecode(data.toString());
// print(finalz);
// get it in the string formati
final check = selected_value_return.fromJson(finalz);
// convert to int for conditions
int check2 = int.parse(check.toString());
//String check2=check.toString();
// checknig where we need to navigate the user
// for the Admin
if (check2==3) {
  Navigator.pushNamed(context, MyRoutes.adminRoute);
  // for the driver
} else if (check2==2) {
  print("Itnto driver");
  Navigator.pushNamed(context, MyRoutes.driverRoute);
} else {
  // for the student
  print("Itnto student");
  Navigator.pushNamed(context, MyRoutes.homeRoute);
}
```

Team Duties:

<u>Mr. Musaab Imran: Scrum Master, developer</u>

- Assign roles, tasks, check and report process.
- Implementation of login and the home page(frontend).
- Documentation.

<u>Mr. Usman Shahid: Administrator, designer, developer</u>

- Implementation of registration and the home page(frontend).
- Implementation of logout functionality.
- Integration of project.

<u>Mr. Ismail Ramzan: User experience engineer, developer, tester</u>

- Connecting project to firebase(backend).
- Unit and integration testing.
- Develop core functions of the app.