

Flutter Application:

Introduction:

Introducing a Flutter application that enables users to classify electrocardiogram (ECG) images with a high degree of accuracy and precision. The application allows users to select an ECG image from their gallery and upload it to a server for classification. The server utilizes an advanced machine learning model to analyze the ECG image and classify it into one of five distinct categories: normal person, COVID patient, myocardial infarction patient, abnormal heartbeats, and patients with history of MI.

The application's classification algorithm is designed to provide users with reliable and actionable insights into ECG data, allowing them to make informed decisions regarding patient care, research, and analysis. The application's user-friendly interface and technology make it an essential tool for medical professionals, researchers, and individuals with a keen interest in ECG analysis.

The application is connected to the machine learning model using Render server, we specifically chose Render because Render is a cloud platform that provides developers with a powerful and flexible infrastructure for building and deploying web applications and services. Render's serverless architecture enables developers to focus on writing code rather than managing servers, allowing for faster development cycles and reduced operational overhead, It also supports a wide range of programming languages, frameworks and libraries, it's platform can handle thousands of requests per second and allows free access.

Frontend:

1)Login:

Figure 1. Login Page.

Figure 2. Login Validation

12:43 28%

Covax

Email

Password

Log In

Don't have an account? [Register Now](#)

12:43 28%

Covax

Email

Email can't be empty

Password

Password can't be empty

Log In

Don't have an account? [Register Now](#)

4:01 100%

Covax

Email

dalia

Email should contain @ and . symbols.

Password

...

Password should be atleast 6 characters long

Log In

Don't have an account? [Register Now](#)

Figure 1. Login Page.
3.Validation

Figure 2. Login Validation

Figure

2)Sign Up:

Figure 4. Sign Upbelow shows the signup screen that the user can use to register a new account during the first time using our application. The user has to enter his/her first and last names, email and password, while *Figure 5.* shows that we have also added the form validation to this page where the user can't register for a new account if any of the required fields is missing. Figure 6. Shows more validation where name can't be less than 6 characters, email can't be written without "@" or "." and finally, password can't be less than 6 characters in length.

12:44 29

Welcome to Covax

Firstname

Lastname

Email

Password

Sign Up

12:44 29

Firstname

Firstname can't be empty

Lastname

Lastname can't be empty

Email

Email can't be empty

Password

Password can't be empty

Sign Up

4:10

Firstname

da

Firstname should be at least 6 characters long

Lastname

ysh

Lastname should be at least 6 characters long

Email

dalia

Email should contain @ and . symbols.

Password

...

Password should be atleast 6 characters long

Sign Up

Figure 4. Sign Up validation

Figure 5. Sign Up Validation

Figure 6. More

3)Model Deployment:

Figure 7. Model Page below shows Diagnosis page where the user can pick an image from phone gallery and this image will appear instead of the app logo, then the user can upload the image to the model and receive result is written on the screen.

Figure 7. Model Page

image shows what the image should look like when the user picks it from gallery.

Figure 8. Example of uploading

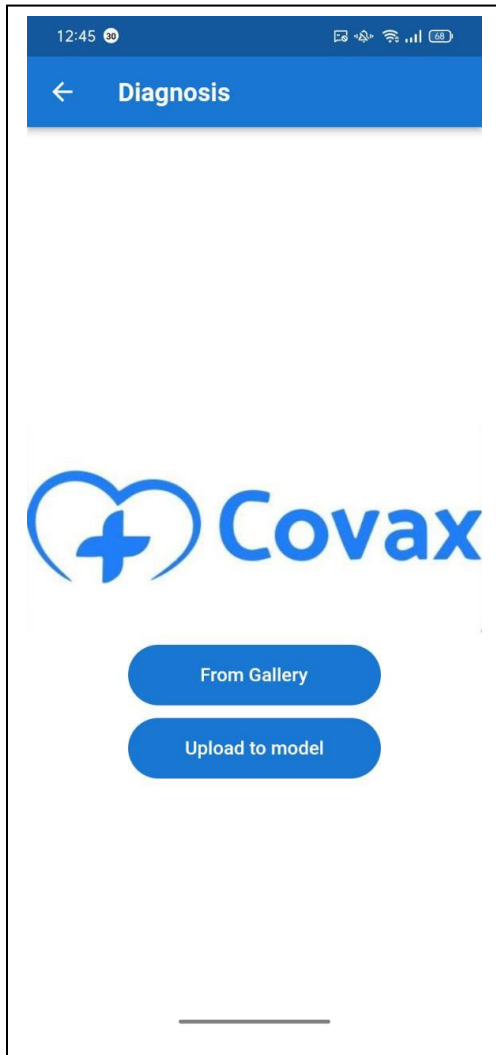


Figure 7. Model Page
image

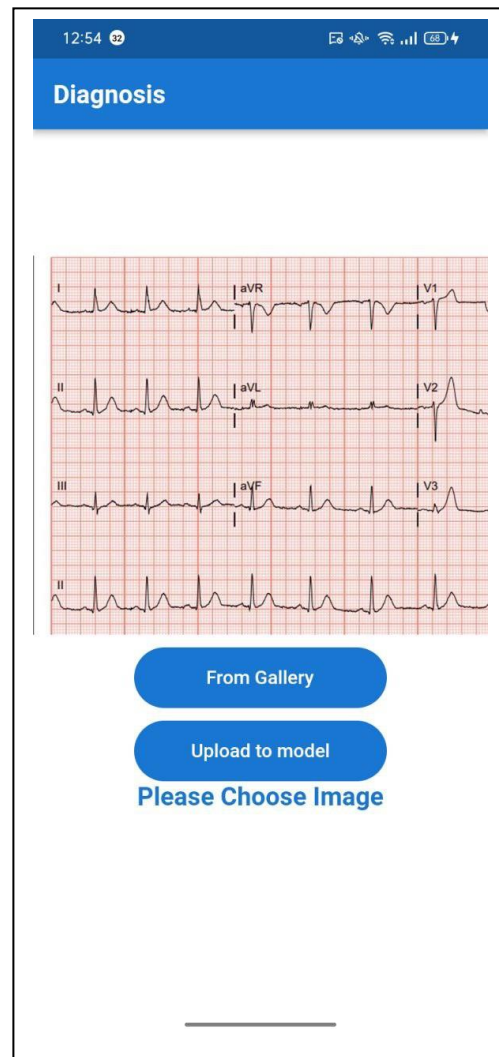


Figure 8. Example of uploading

4)Test Cases:

The figures below show the result of the model when it was tested on different cases from the 5 classes we mentioned before.

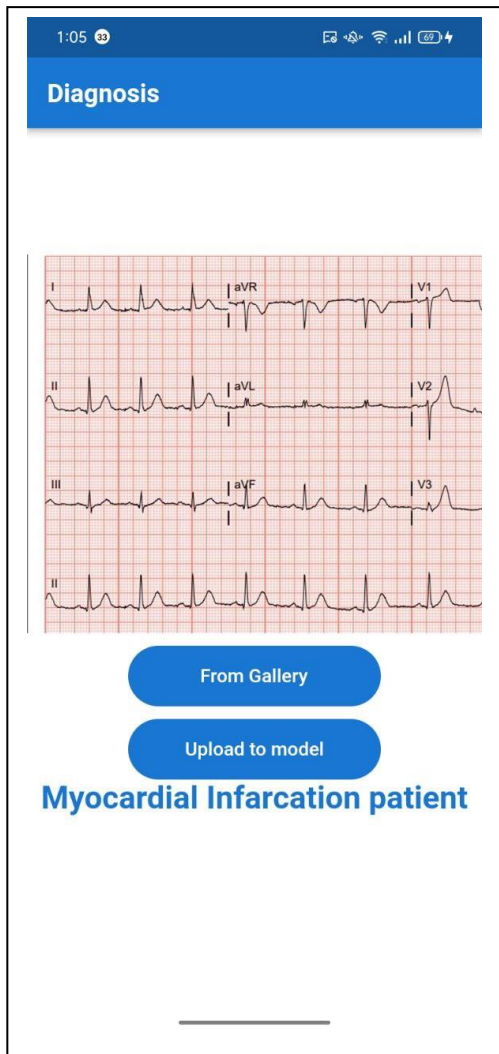


Figure 9. Myocardial Infarction case

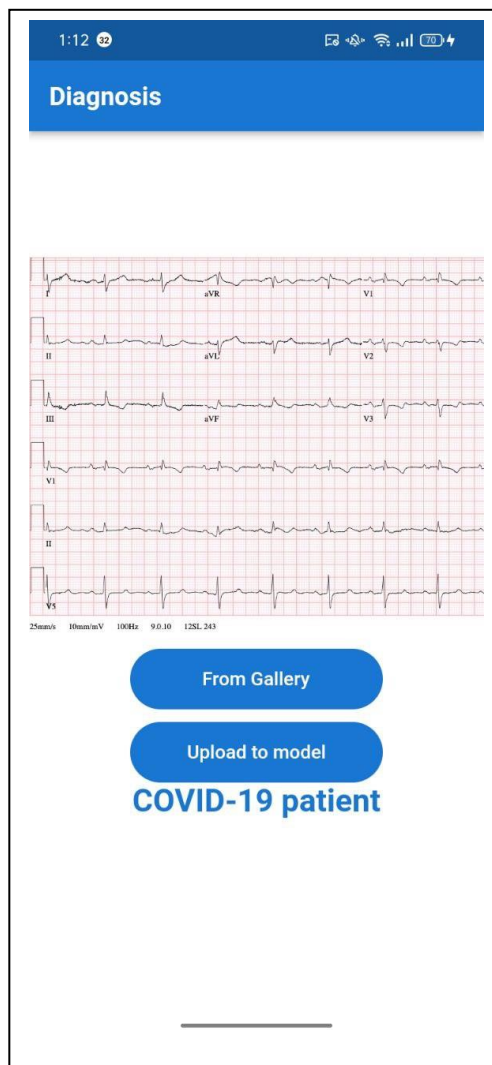


Figure 10. COVID-19 case

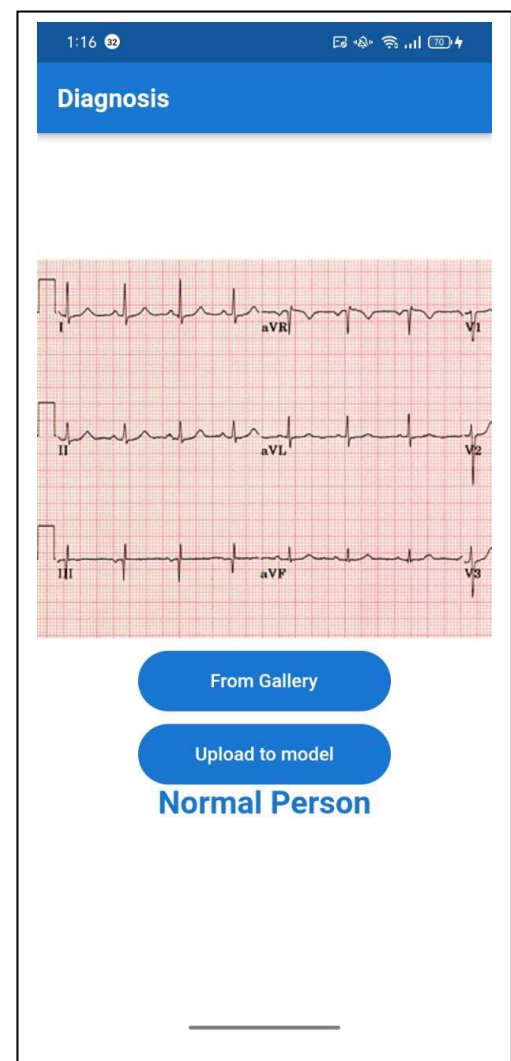


Figure 11.

App Authentication:

For authentication we used *Shared Preferences* library. *Shared preferences* is a powerful and convenient tool for managing local data storage in Flutter applications. It provides a

simple and efficient way to store and retrieve data that can be easily accessed throughout the application. One of the main benefits of using shared preferences is that it allows developers to store small amounts of data, such as user preferences, authentication tokens, and app settings, without the need for a complex database setup. This makes it a great choice for applications that require simple data storage that can be accessed quickly and easily. Another advantage is that Shared Preferences is well-documented and can be integrated in any Flutter project easily, making it easy for developers to add, modify and retrieve data as needed. it's also a local data storage tool so this will help the user to keep access to the application even if there was a problem with the server connection, where it depends on the idea of caching frequently used data so that, the user doesn't have to enter his/her data again every time he/she uses the app.

Test Cases:

1) The user tries to login while he/she doesn't even have an account

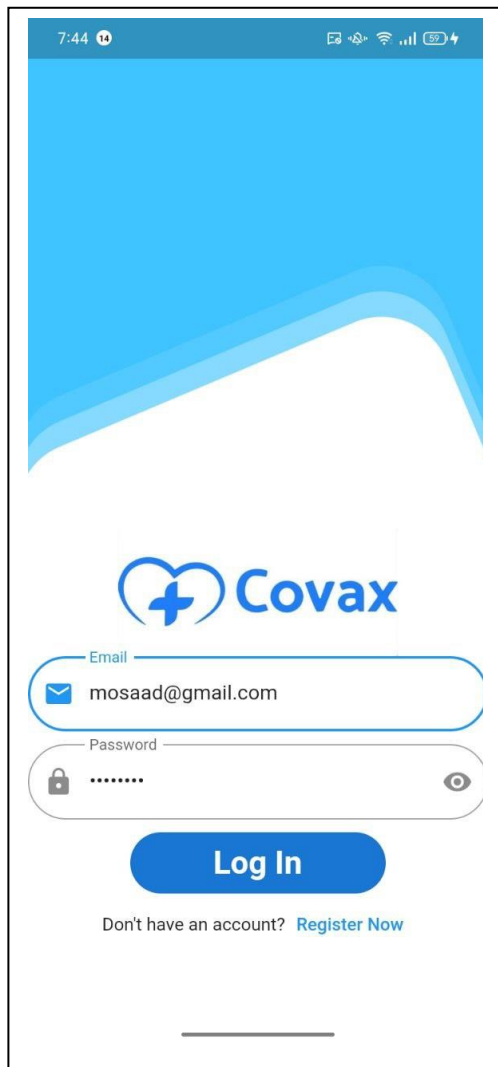


Figure 12.Login without having an account before.

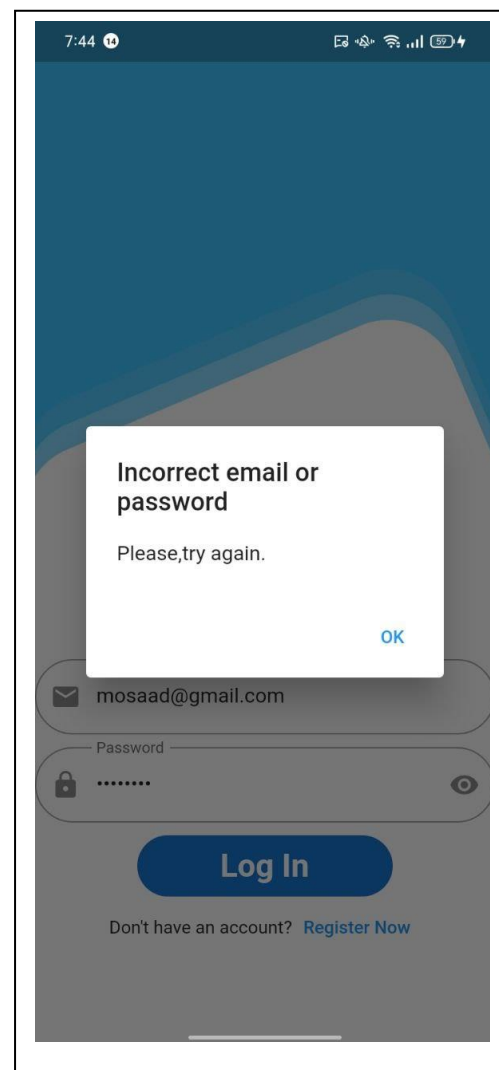


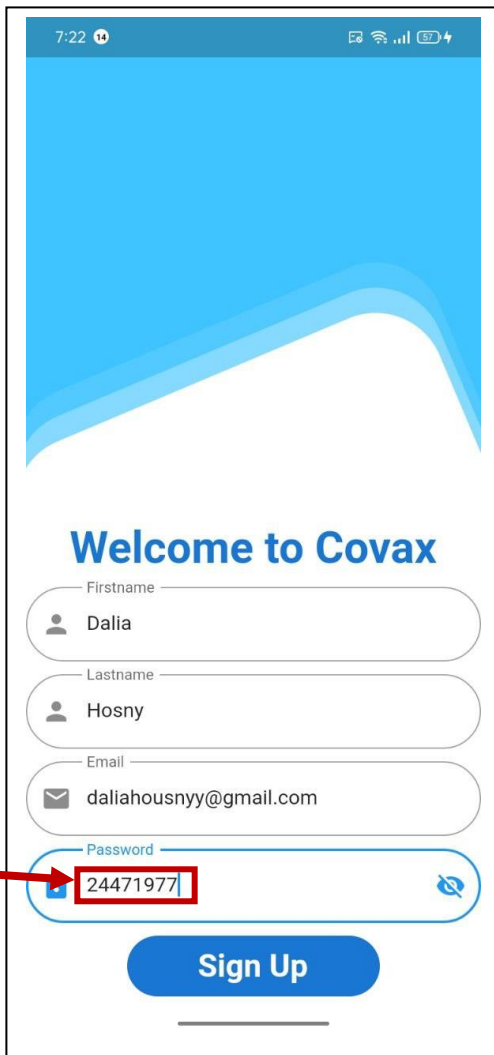
Figure 13.Authentication Error.

The two figures above shows the first case where *Figure 12.Login* without having an account before.

Figure 13.Authentication Error. shows a user who uses the app for the first time and tries to login without registering his data before, so as shown in *Figure 12.Login* without having an account before.

Figure 13.Authentication Error. a dialog box containing an error appeared on the screen because the data the user entered doesn't match any of that stored in app data.

- 2) The user registers an account and data is stored but during the next time using the app she enters an incorrect password.



7:22 14

Welcome to Covax

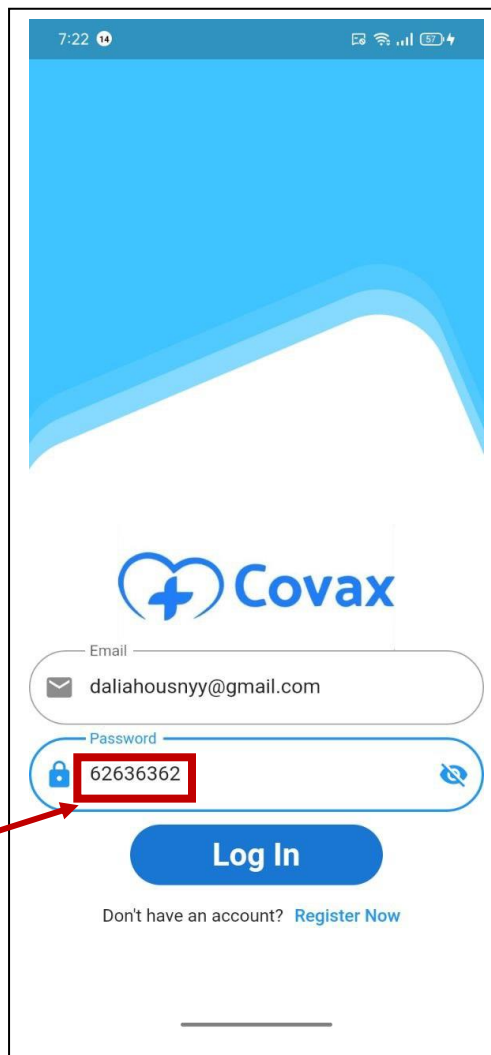
Firstname
Dalia

Lastname
Hosny

Email
daliahousnyy@gmail.com

Password
24471977

Sign Up



7:22 14

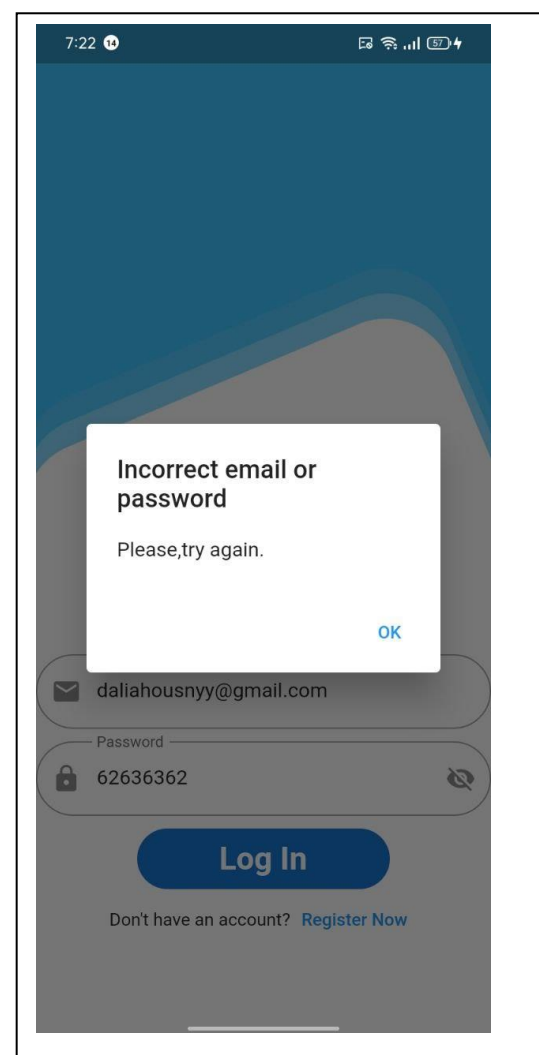
Covax

Email
daliahousnyy@gmail.com

Password
62636362

Log In

Don't have an account? [Register Now](#)



7:22 14

Incorrect email or password

Please,try again.

OK

daliahousnyy@gmail.com

Password
62636362

Log In

Don't have an account? [Register Now](#)

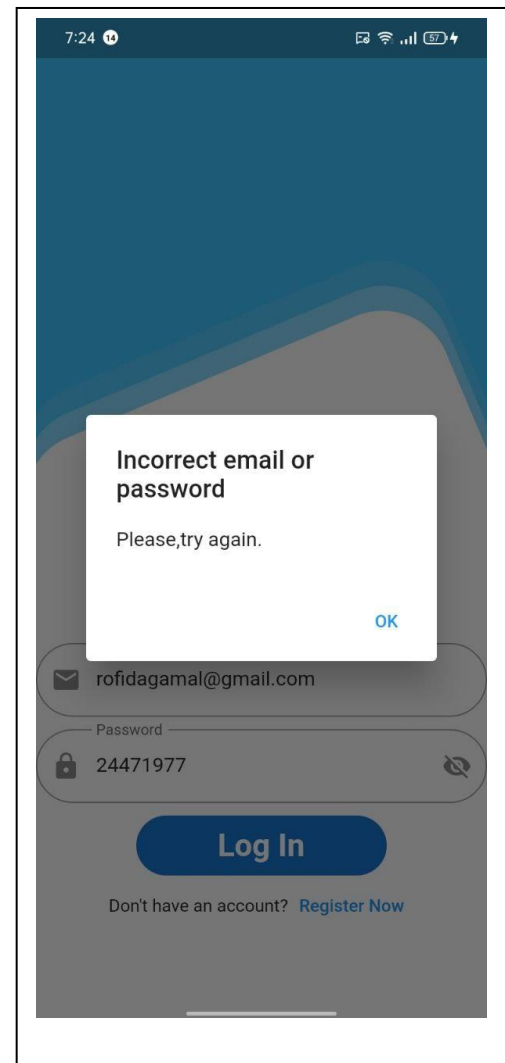
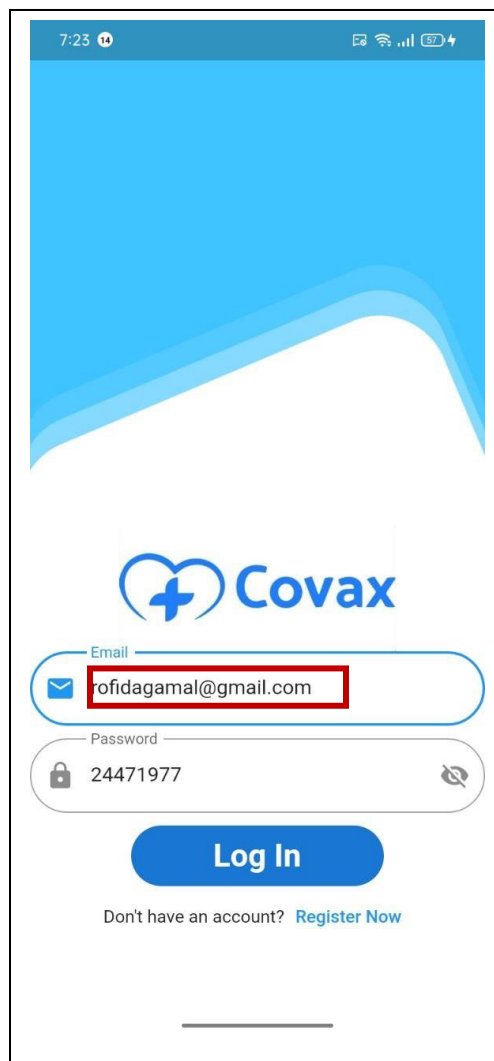
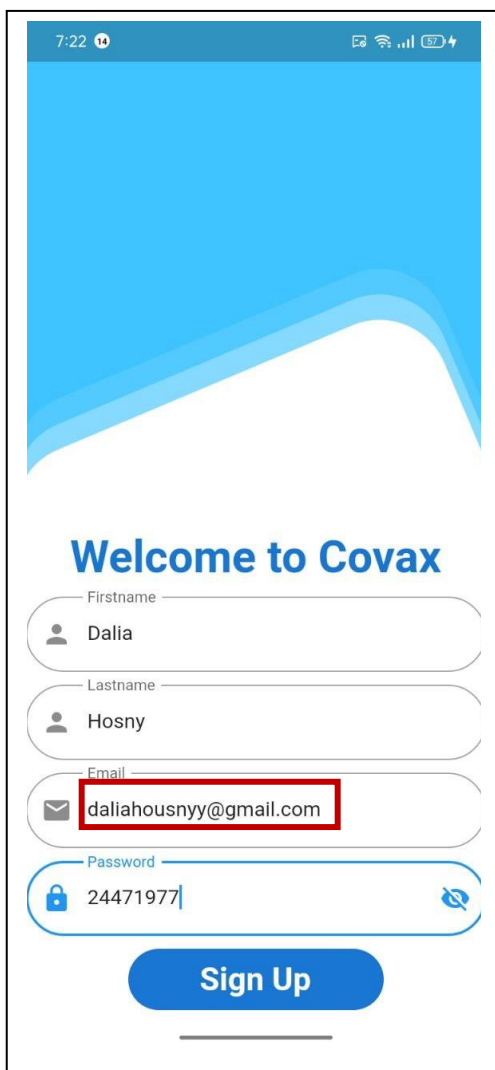
*Figure 14. Signing Up
Error*

Figure 15. Wrong password.

Figure 16. Authentication

The 3 figures above explain the second case where, the user enters a wrong password when logging in to her account. As shown in *Figure 14* the user signs up for the application by entering her email and password but in *Figure 15* she enters a wrong password that doesn't match the one she entered the first time and stored in our app's database so in *Figure 16* a dialog box containing an error appears asking the user to try again but by typing the correct password this time.

3) The user enters an incorrect email.



*Figure 17. Signing up
19. Authentication error*

Figure 18. Wrong Email.

Figure

The 3 figures above explain the third case where, the user enters a wrong email when logging in to her account. As shown in *Figure 17* the user signs up for the application by entering her email and password but in *Figure 18* she enters a wrong email that doesn't match the one she entered the first time and stored in our app's database so in *Figure 19* a dialog box containing an error appears asking the user to try again but by typing the correct email this time.