Password Manager

Project by:
Rohit Pai: 2018130033
Harsh Sandesara: 2018130045
Vishal Salvi: 2019230069

Sardar Patel Institute of Technology
S.E. Comps, Academic Year 2019-20

Index:

About the project

OneKey

OneKey is a consumer-oriented password manager which is used to store and manage user passwords so that users can login to online websites and apps without having to remember login information. This app automatically detects input fields and fills all the details so that users don't forget their credentials.

Why use this app?

In today's world, information is key. However, with so much information to remember, it is easy to forget trivial things such as passwords. This is where our app comes in handy. With OneKey, we aim to lessen some of the forgettable, but extremely important user information: passwords. By using this app, users can stop worrying about having to remember multiple login IDs and passwords and focus on more important things, leaving the job to this app.

Features

- No more forgetting passwords:
  Store and manage URLs, usernames, and passwords easily. You can store your own password or generate a random password using the Generator button.
- Create Notes on the go:
  With the Notes feature, you can create secure memos and access them anytime.
- One account, multiple devices:
  Access your account on any device. You need only remember one master password and the app takes care of all your other passwords for you.
- Security first:
  All information you store is encrypted so that only you have access to your information. This makes it safe, secure, and reliable.

Software used

For the app:

- **Android Studio:**
  Android Studio is the main tool used to develop, debug and run the app. The app has been developed using JAVA, which is part of our curriculum. The ability to create and modify front- and back-end content easily is the reason we have used Android Studio.

- **Autofill Framework:**
  One of the main features of our app is Autofill Service. The communication between our app and the autofill service is managed by the Autofill Framework.

For the database:

- **Firebase (cloud Firestore):**
  Firebase is Google's mobile database platform that helps store user data. Since it is built on Google infrastructure, it becomes easy to integrate it with our Android app. It also makes it easy to store and sync user data at a global scale.

OOP Concepts:

- Multithreading:
    - Multithreading is used to execute multiple threads concurrently. We have used this concept to create the Splash Screen of our app. The Runnable interface is used to create a thread which starts the next activity using intents. The Handler object method postDelayed() is used to delay the above task long enough to make the Splash Screen visible to users.
- Polymorphism:
    - Polymorphism is the ability of an object to take on many forms. Essentially, we have achieved runtime polymorphism by overriding predefined methods in the activity class such as onStart(), onCreate(), onResume(), etc. We use the super keyword while overriding the methods so that we can redefine the method and modify it according to our use.
- Encapsulation:
    - Encapsulation is used to hide the values or states of an object inside a class, preventing unauthorized parties' direct access to them. Since security lies at the centre of our app, encapsulation is one of the most important concepts used in our project. All sensitive data like passwords and user IDs is encapsulated in order to provide utmost privacy and security to users.
- Inheritance:
    - Inheritance is the mechanism by which one class is allowed to inherit the features of another class. We have used inheritance several times in our project to extend classes such as AppCompatActivity, which finds importance in the making of the app, and AutofillService, which is used to automatically fill in text fields outside of the app.

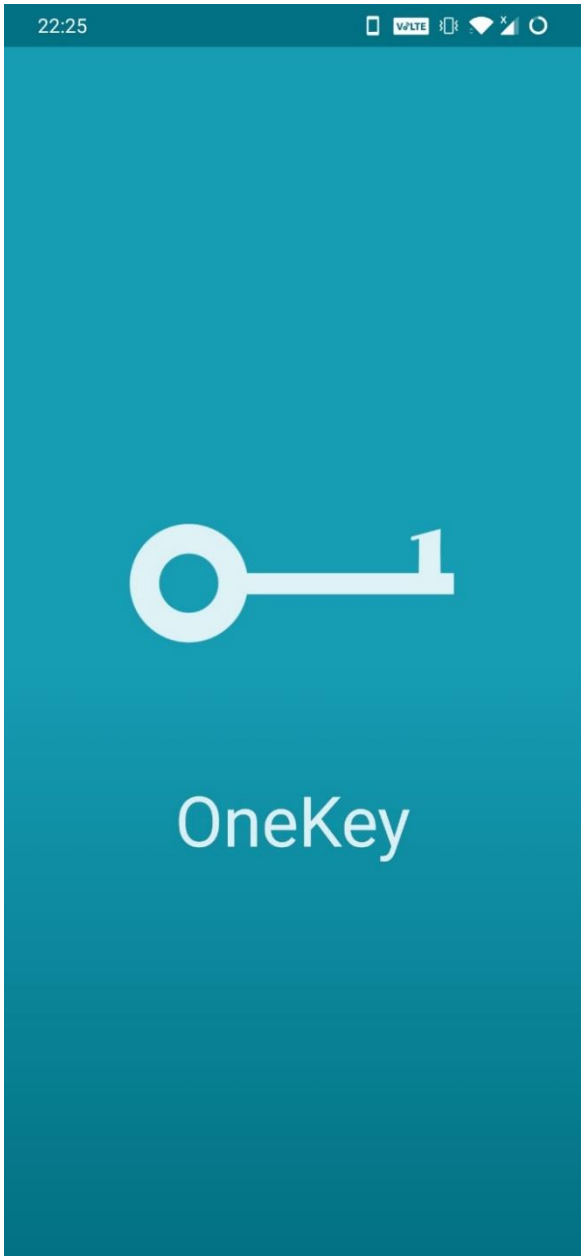
Code:

```
package com.example.onekey;

import android.app.assist.AssistStructure;
import android.os.CancellationSignal;
import android.service.autofill.AutofillService;
import android.service.autofill.Dataset;
import android.service.autofill.FillCallback;
import android.service.autofill.FillContext;
import android.service.autofill.FillRequest;
import android.service.autofill.FillResponse;
import android.service.autofill.SaveCallback;
import android.service.autofill.SaveInfo;
import android.service.autofill.SaveRequest;
import android.util.Log;
import android.view.autofill.AutofillId;
import android.view.autofill.AutofillManager;
import android.view.autofill.AutofillValue;
import android.widget.RemoteViews;

import androidx.annotation.NonNull;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;


import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
import com.google.firebase.firestore.Source;

import java.util.ArrayList;
import java.util.List;

import static android.view.View.AUTOFILL_HINT_PASSWORD;
import static android.view.View.AUTOFILL_HINT_USERNAME;
import static com.example.onekey.EncryptDecryptString.decrypt;

public class MyAutofillService extends AutofillService {
    private static final String TAG = "MyAutofillService";
    private AutofillId usernameAutofillId;
    private AutofillId passwordAutofillId;
    private ArrayList<Password> passwordArray = new ArrayList<>();
    private FirebaseAuth mAuth;

    @Override
    public void onFillRequest(@NonNull FillRequest fillRequest, @NonNull
CancellationSignal cancellationSignal, @NonNull FillCallback fillCallback) {
        Log.d(TAG, "onFillRequest:");
        List<FillContext> context = fillRequest.getFillContexts();
        AssistStructure structure = context.get(context.size() -
1).getStructure();

        traverseStructure(structure);

        if (usernameAutofillId != null && passwordAutofillId != null) {
```

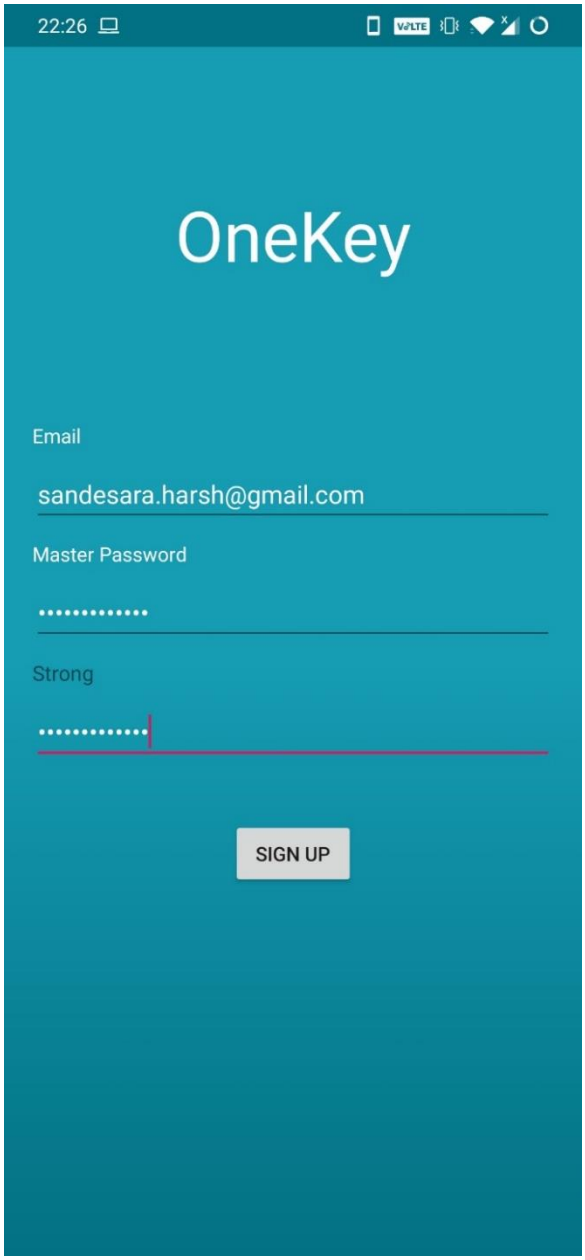
```
            Source source = Source.CACHE;
            mAuth = FirebaseAuth.getInstance();

            if (mAuth.getCurrentUser() != null) {
                FirebaseFirestore db = FirebaseFirestore.getInstance();

                db.collection("Users").document(mAuth.getCurrentUser().getEmail())
                        .collection("URL")
                        .get(source)
                        .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
                            @Override
                            public void onComplete(@NonNull Task<QuerySnapshot>
task) {
                                if (task.isSuccessful()) {
                                    for (QueryDocumentSnapshot document :
task.getResult()) {
                                        Log.d(TAG, document.getId() + " => " +
document.getData());
                                        Password password =
document.toObject(Password.class);

password.setUrl(decrypt(password.getUrl()));

password.setUsername(decrypt((password.getUsername())));

password.setPassword(decrypt((password.getPassword())));
                                        passwordArray.add(password);
                                    }
                                } else {
                                    Log.d(TAG, "Error getting documents: ",
task.getException());
                                }
                            }
                        });

                Log.d(TAG, "onFillRequest: Autofill");
                ParsedStructure parsedStructure = new
ParsedStructure(usernameAutofillId, passwordAutofillId);

                //UserData userData = fetchUserData(parsedStructure);

                if (!passwordArray.isEmpty()) {
                    Log.d(TAG, "onFillRequest: Hello");
                    FillResponse.Builder fillResponse = new
FillResponse.Builder();

                    for (int i = 0; i < passwordArray.size(); i++) {
                        RemoteViews usernamePresentation = new
RemoteViews(getPackageName(), R.layout.test_autofill_menu_item);

usernamePresentation.setImageViewResource(R.id.autofill_icon,
R.drawable.password_icon_black);
                        usernamePresentation.setTextViewText(R.id.autofill_text,
                                " " + passwordArray.get(i).getUrl() + "\n" + " " +
passwordArray.get(i).getUsername());

                        fillResponse.addDataset(new Dataset.Builder()
                                .setValue(parsedStructure.usernameId,
```

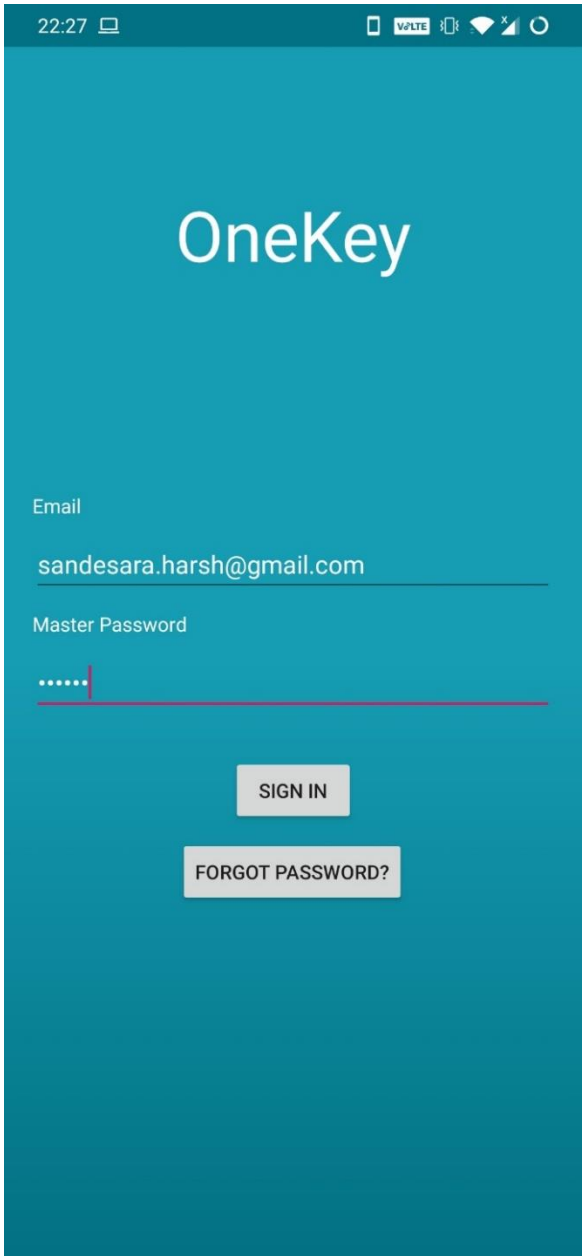
```
AutofillValue.forText(passwordArray.get(i).getUsername()), usernamePresentation)
                                .setValue(parsedStructure.passwordId,

AutofillValue.forText(passwordArray.get(i).getPassword()), usernamePresentation)
                                .build());
                    }

                    AutofillId[] autofillIds = new
AutofillId[]{usernameAutofillId, passwordAutofillId};
                    SaveInfo saveInfo = new
SaveInfo.Builder(SaveInfo.SAVE_DATA_TYPE_GENERIC, autofillIds).build();

                    if(saveInfo != null) {
                        Log.d(TAG, "onFillRequest: SaveInfo");
                        fillResponse.setSaveInfo(saveInfo);
                        Log.d(TAG, "onFillRequest: " +
usernameAutofillId.toString() + " " + passwordAutofillId.toString() + " " +
saveInfo.toString());
                        Log.d(TAG, "onFillRequest: " + autofillIds[0].toString() +
" " + autofillIds[1].toString());
                    }

                    fillCallback.onSuccess(fillResponse.build());
                }
            }

        }
    }

    class ParsedStructure {
        AutofillId usernameId;
        AutofillId passwordId;

        public ParsedStructure(AutofillId usernameId, AutofillId passwordId) {
            this.usernameId = usernameId;
            this.passwordId = passwordId;
        }
    }

    class UserData {
        String username;
        String password;
    }

    public void traverseStructure(AssistStructure structure) {
        int nodes = structure.getWindowNodeCount();

        for (int i = 0; i < nodes; i++) {
            AssistStructure.WindowNode windowNode = structure.getWindowNodeAt(i);
            AssistStructure.ViewNode viewNode = windowNode.getRootViewNode();
            traverseNode(viewNode);
        }
    }

    public void traverseNode(AssistStructure.ViewNode viewNode) {
        if (viewNode.getAutofillHints() != null &&
viewNode.getAutofillHints().length > 0) {
            String[] autofill = viewNode.getAutofillHints();
```

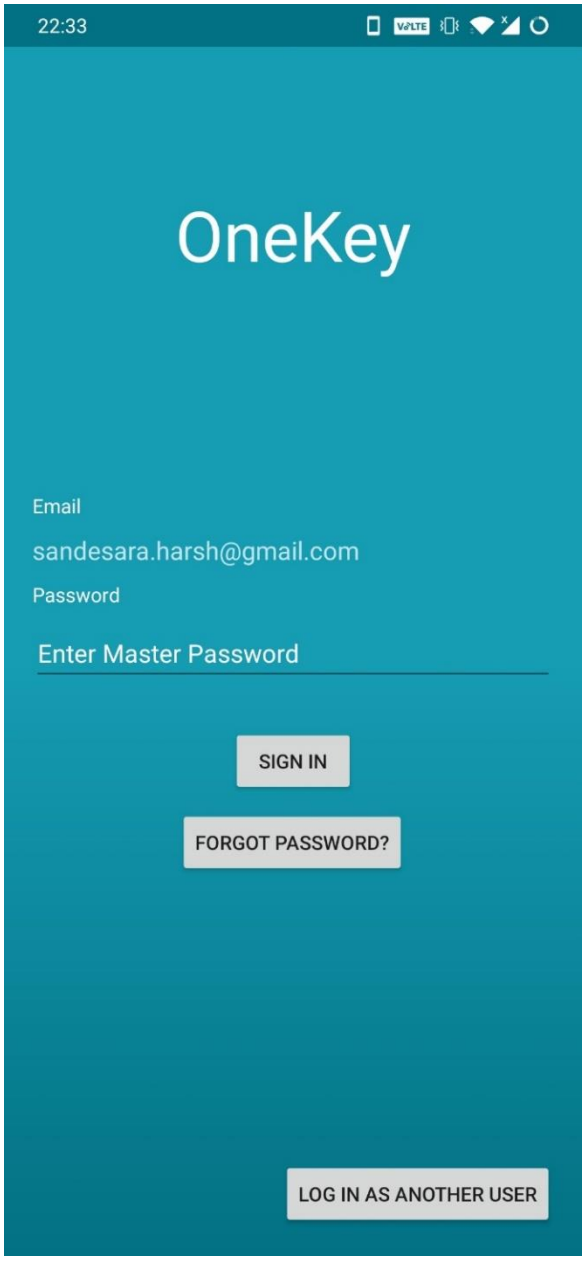
```
            for (String s : autofill) {
                if (s.equals(AUTOFILL_HINT_USERNAME)) {
                    usernameAutofillId = viewNode.getAutofillId();
                }
                if (s.equals(AUTOFILL_HINT_PASSWORD)) {
                    passwordAutofillId = viewNode.getAutofillId();
                }
            }
        } else {
            String s = viewNode.getHint();
            if (s != null) {
                if (s.contains("Email") || s.contains("email") ||
s.contains("Username") || s.contains("username")) {
                    usernameAutofillId = viewNode.getAutofillId();
                }
                if (s.contains("Password") || s.contains("password")) {
                    passwordAutofillId = viewNode.getAutofillId();
                }
            }
        }

        for (int i = 0; i < viewNode.getChildCount(); i++) {
            AssistStructure.ViewNode childNode = viewNode.getChildAt(i);
            traverseNode(childNode);
        }
    }

    @Override
    public void onSaveRequest(@NonNull SaveRequest saveRequest, @NonNull
SaveCallback saveCallback) {
        Log.d(TAG, "onSaveRequest: ");
        List<FillContext> context = saveRequest.getFillContexts();
        AssistStructure structure = context.get(context.size() -
1).getStructure();

        traverseStructure(structure);

        saveCallback.onSuccess();
    }

    @Override
    public void onConnected() {
        Log.d(TAG, "onFillRequest: onConnected");
    }

    @Override
    public void onDisconnected() {
        Log.d(TAG, "onFillRequest: onDisconnected");
    }
}
```

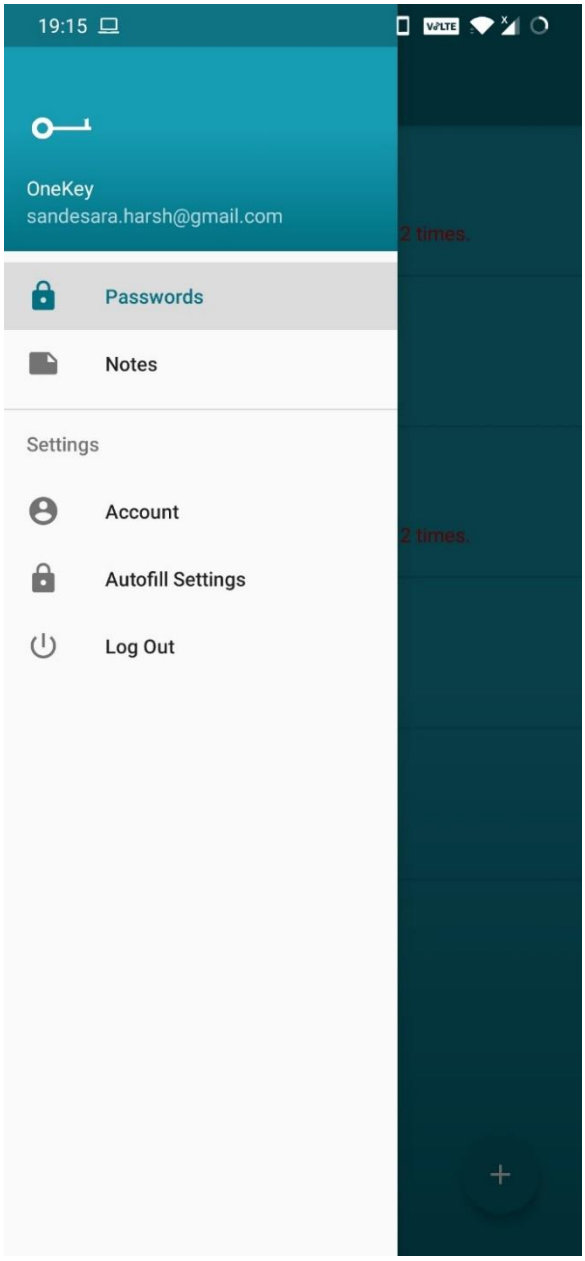

22:25
OneKey

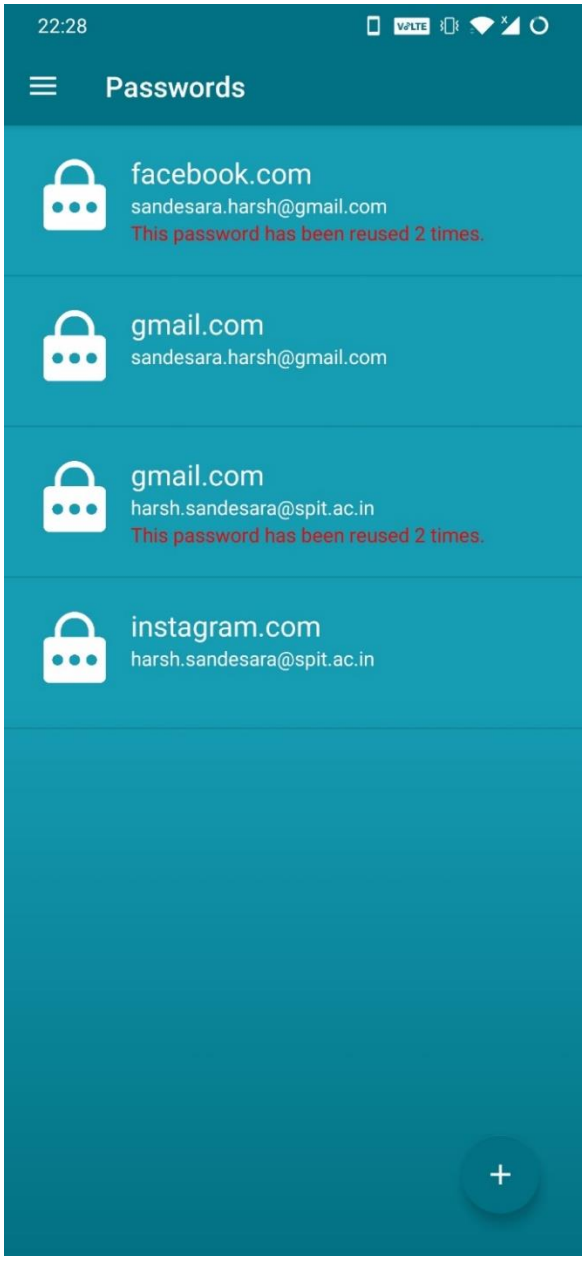

22:26
OneKey
SIGN IN
SIGN UP

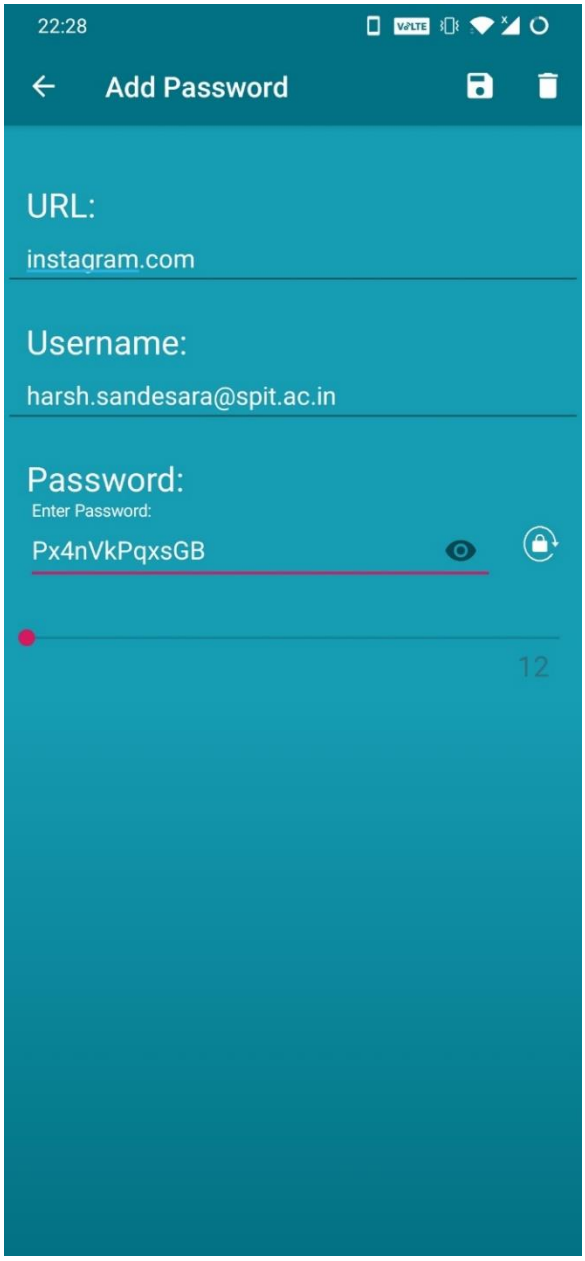

22:26
OneKey
Email
sandesara.harsh@gmail.com
Master Password
Strong
SIGN UP

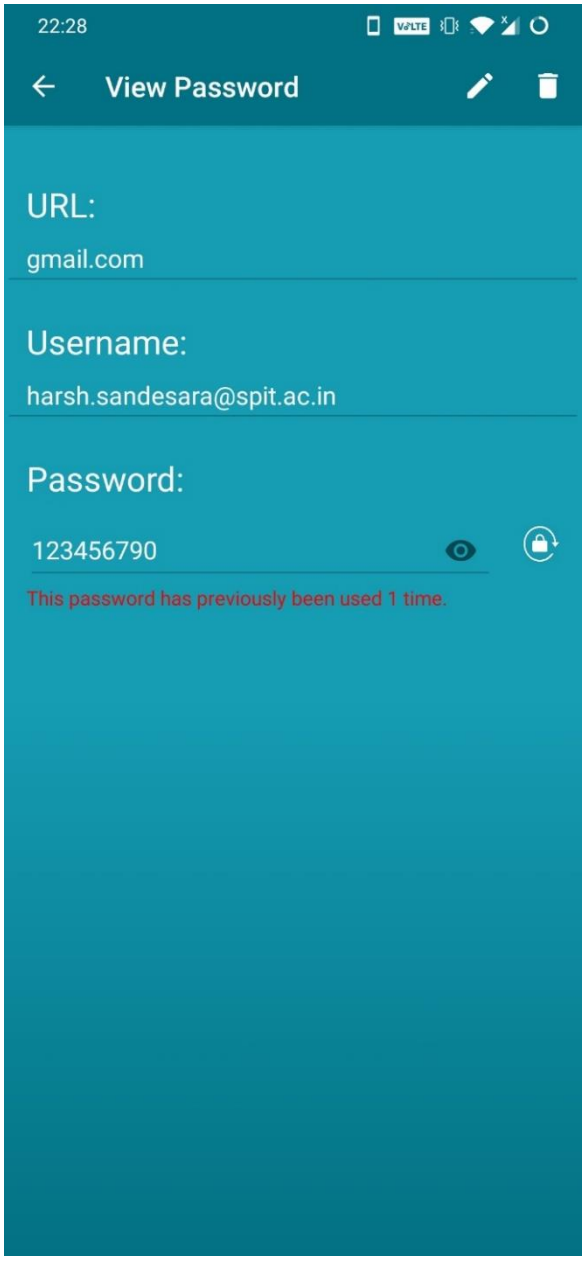

22:27
OneKey
Email
sandesara.harsh@gmail.com
Master Password
SIGN IN
FORGOT PASSWORD?

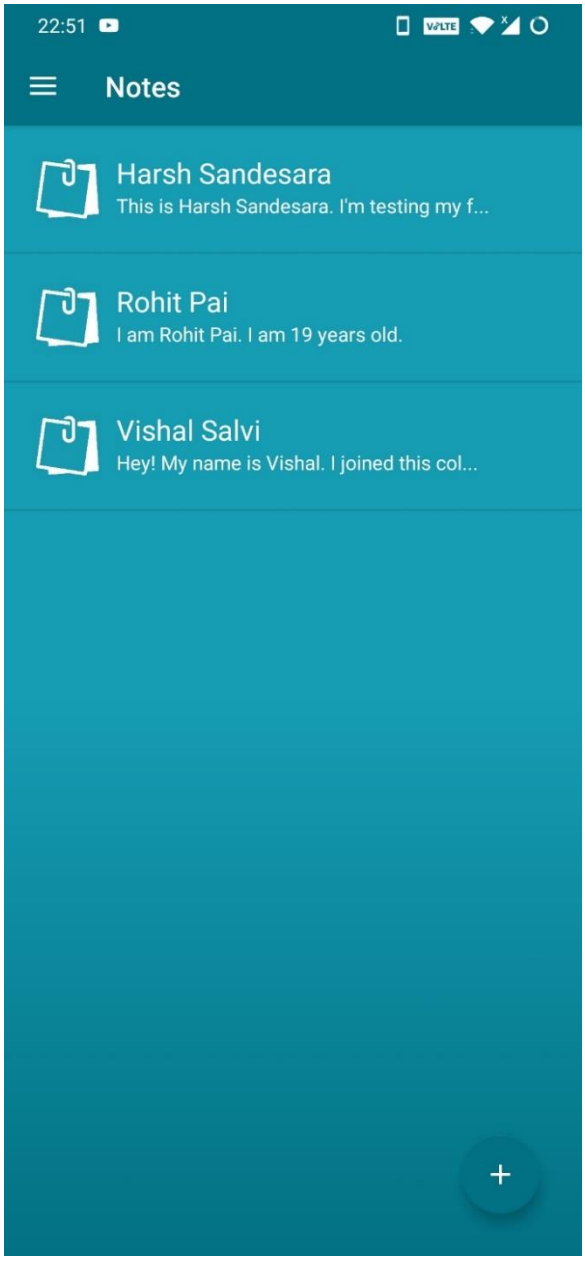

22:33
OneKey
Email
sandesara.harsh@gmail.com
Password
Enter Master Password
SIGN IN
FORGOT PASSWORD?
LOG IN AS ANOTHER USER


19:15
OneKey
sandesara.harsh@gmail.com
Passwords
Notes
Settings
Account
Autofill Settings
Log Out
2 times.
2 times.


22:28
Passwords
facebook.com
sandesara.harsh@gmail.com
This password has been reused 2 times.
gmail.com
sandesara.harsh@gmail.com
gmail.com
harsh.sandesara@spit.ac.in
This password has been reused 2 times.
instagram.com
harsh.sandesara@spit.ac.in


22:28
Add Password
URL:
instagram.com
Username:
harsh.sandesara@spit.ac.in
Password:
Enter Password:
Px4nVkPqxsGB
12

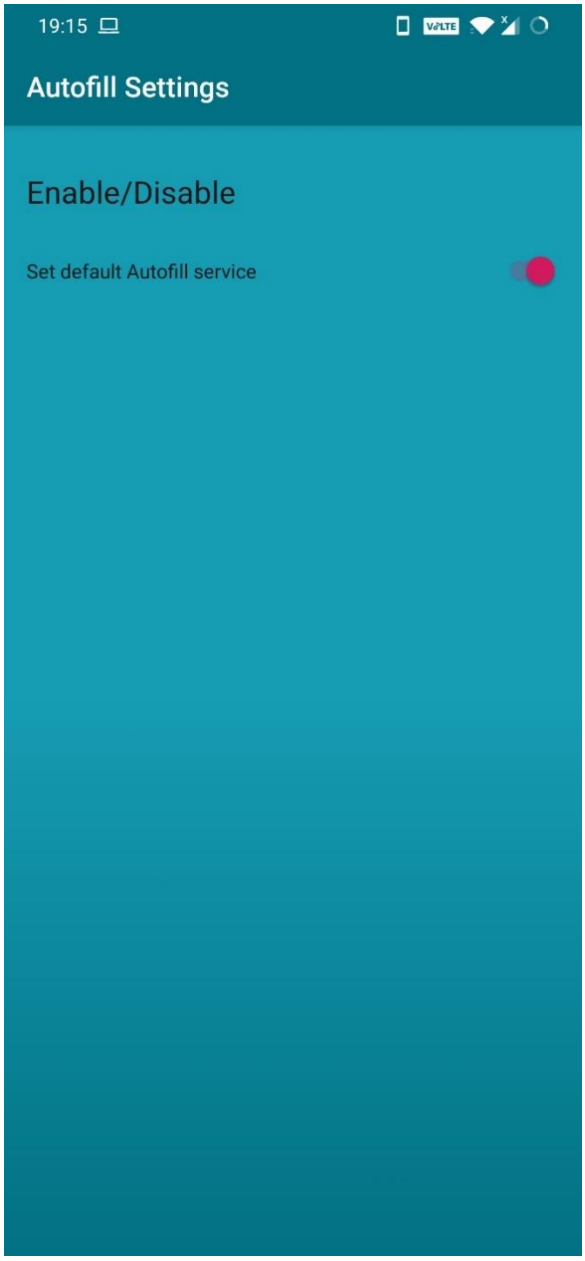

22:28
View Password
URL:
gmail.com
Username:
harsh.sandesara@spit.ac.in
Password:
123456790
This password has previously been used 1 time.

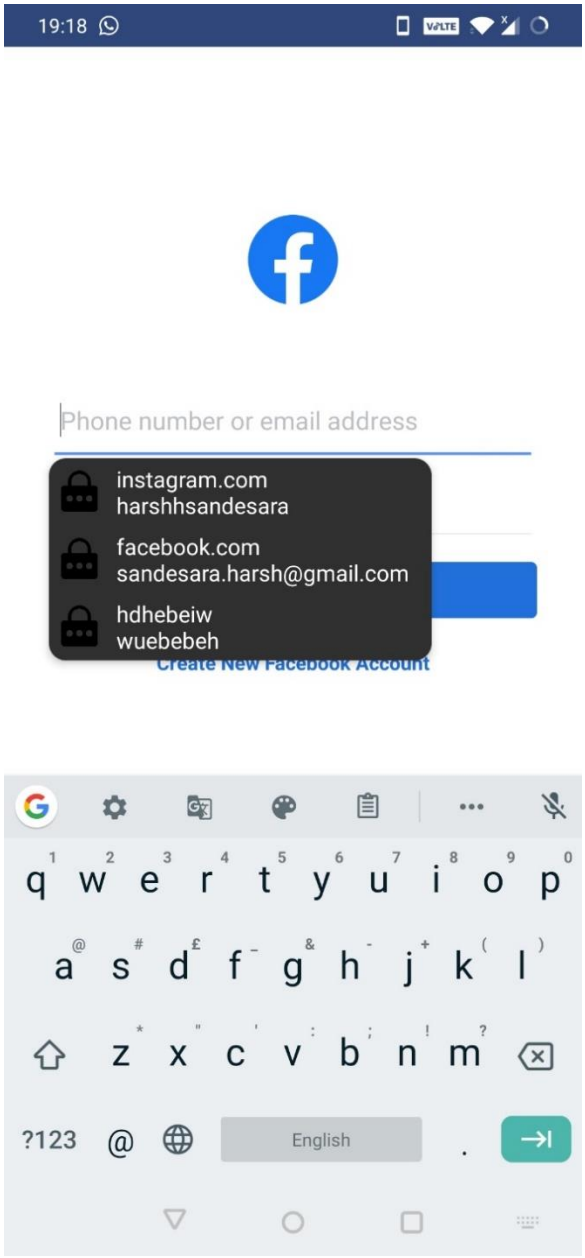

22:51
Notes
Harsh Sandesara
This is Harsh Sandesara. I'm testing my f...
Rohit Pai
I am Rohit Pai. I am 19 years old.
Vishal Salvi
Hey! My name is Vishal. I joined this col...


22:51
Add Note
Add a Title...
What's On Your Mind?


22:29
Edit Note
Harsh Sandesara
This is Harsh Sandesara. I'm testing my first Android project.


22:31
Account
Reset Password
RESET PASSWORD


19:15
Autofill Settings
Enable/Disable
Set default Autofill service


19:18
Phone number or email address
instagram.com
harshhsandesara
facebook.com
sandesara.harsh@gmail.com
hdhebeiw
wuebebeh
Create New Facebook Account
English

Conclusion:

- JAVA is an important language for android app developers. It is used widely for programming and app development.
- While doing this project, we have come across several useful functions and advantages of JAVA.
- Developing this app alongside the curriculum-specified Object-Oriented Programming has been advantageous to us in that we are now better versed with several key concepts of OOP. It has also strengthened our debugging and collaborative skills.

References:

www.github.com
developer.android.com
www.geeksforgeeks.org
www.stackoverflow.com