**Mobile Application Development Mini Project**

**Ebenezer Isaac - 2020178014**

**Aim:**

To create a password managing application with industry standard encryption of AES GCM and Google KMS Envelope Encryption.

**Challenge:**

The application needs to be build in such a way that even if application logic is leaked, the data cannot be compromised

**Procedure:**

1. Create a new Android Studio Project
2. Design necessary layouts with the help of xml files.
3. Create Firebase Project and enable Firestore and Firebase Authentication with Google Sign-In.
4. Create Google Cloud Project and obtain Google Key Management System API Key.
5. Create a REST API using NodeJS to facilitate Envelope Encryption for KeysetHandle of AES GCM
6. Create relevant classes to encrypt and decrypt the information securely.

**Tools Used:**

Android Studio, Google Cloud Platform, Google Key Management System, Google Tink, Firebase Firestore, Firebase Authentication, Google reCaptcha v2, NodeJS, Express JS, JSON, Sublime

**Guide:**

Mr. H. Riasudheen

**Android Application:**

Manager/PasswordItem.java

package com.mycrolinks.passwdmgr.manager;  
  
import android.annotation.SuppressLint;  
  
import org.json.JSONObject;  
  
import java.io.Serializable;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
  
public class PasswordItem implements Serializable {  
 JSONObject passwordItem;  
  
 public PasswordItem(String title, String password, String timestamp) {  
 passwordItem = new JSONObject();  
 try {  
 passwordItem.put("title", title);  
 passwordItem.put("password", password);  
 passwordItem.put("timestamp", timestamp);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public PasswordItem(String title, String password) {  
 passwordItem = new JSONObject();  
 try {  
 Date today = Calendar.*getInstance*().getTime();  
 @SuppressLint("SimpleDateFormat") DateFormat df = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
 passwordItem.put("title", title);  
 passwordItem.put("password", password);  
 passwordItem.put("timestamp", df.format(today));  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public PasswordItem(JSONObject passwordItem) {  
 this.passwordItem = passwordItem;  
 }  
  
 public String getTitle() {  
 try {  
 return passwordItem.getString("title");  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public String getPassword() {  
 try {  
 return passwordItem.getString("password");  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public String getTimestamp() {  
 try {  
 return passwordItem.getString("timestamp");  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public JSONObject getJSONObject() {  
 return passwordItem;  
 }  
}

Manager/PasswordItemList.java

package com.mycrolinks.passwdmgr.manager;  
  
import com.google.crypto.tink.KeysetHandle;  
import com.google.firebase.firestore.DocumentReference;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.util.Arrays;  
  
public class PasswordItemList {  
  
 JSONArray passwordItemList;  
 String documentName;  
 KeysetHandle keysetHandle;  
  
 public PasswordItemList() {  
 passwordItemList = new JSONArray();  
 PasswordItem dummy = new PasswordItem("Dummy", "example password");  
 add(dummy);  
 }  
  
 public PasswordItemList(byte[] passwordItemList) {  
 try {  
 this.passwordItemList = new JSONArray(new String(passwordItemList));  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void delete(int index) {  
 passwordItemList.remove(index);  
 saveToDB();  
 }  
  
 public void add(PasswordItem passwordItem) {  
 passwordItemList.put(passwordItem.getJSONObject());  
 saveToDB();  
 }  
  
 public void update(int index, PasswordItem passwordItem) {  
 try {  
 passwordItemList.remove(index);  
 passwordItemList.put(index, passwordItem.getJSONObject());  
 saveToDB();  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public byte[] toByteArray() {  
 return passwordItemList.toString().getBytes();  
 }  
  
  
 public int getSize() {  
 return passwordItemList.length();  
 }  
  
 public PasswordItem getPasswordItem(int index) {  
 try {  
 return new PasswordItem((JSONObject) passwordItemList.get(index));  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public void saveToDB() {  
 if (documentName != null) {  
 byte[] passwordListByteArray = toByteArray();  
 byte[] encryptedData = Encryptor.*encrypt*(keysetHandle, passwordListByteArray);  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
 DocumentReference documentReference = db.collection("root").document(documentName);  
 documentReference  
 .update("data", Arrays.*toString*(encryptedData))  
 .addOnSuccessListener(aVoid -> System.*out*.println("DocumentSnapshot successfully updated!"))  
 .addOnFailureListener(e -> System.*out*.println("Error updating document" + e.getMessage()));  
 }  
 }  
  
 public void setDocumentName(String documentName) {  
 this.documentName = documentName;  
 }  
  
 public String getDocumentName() {  
 return documentName;  
 }  
  
 public void setKeysetHandle(KeysetHandle keysetHandle) {  
 this.keysetHandle = keysetHandle;  
 }  
  
}

Manager/Encryptor.java

package com.mycrolinks.passwdmgr.manager;  
  
import com.google.common.primitives.Bytes;  
import com.google.crypto.tink.Aead;  
import com.google.crypto.tink.CleartextKeysetHandle;  
import com.google.crypto.tink.JsonKeysetReader;  
import com.google.crypto.tink.JsonKeysetWriter;  
import com.google.crypto.tink.KeyTemplates;  
import com.google.crypto.tink.KeysetHandle;  
import com.google.crypto.tink.aead.AeadConfig;  
  
import java.io.ByteArrayInputStream;  
import java.io.ByteArrayOutputStream;  
import java.io.IOException;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.security.GeneralSecurityException;  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
  
public class Encryptor {  
  
 private static String bytesToHexString(byte[] bytes) {  
 StringBuilder sb = new StringBuilder();  
 for (byte aByte : bytes) {  
 String hex = Integer.*toHexString*(0xFF & aByte);  
 if (hex.length() == 1) {  
 sb.append('0');  
 }  
 sb.append(hex);  
 }  
 return sb.toString();  
 }  
  
 private static String hash(String password, String salt) {  
 MessageDigest digest;  
 String hash;  
 try {  
 digest = MessageDigest.*getInstance*("SHA-256");  
 digest.update((password + salt).getBytes());  
 hash = *bytesToHexString*(digest.digest());  
 return hash;  
 } catch (NoSuchAlgorithmException e1) {  
 e1.printStackTrace();  
 }  
 return null;  
 }  
  
 public static String hash\_wrapper(String password) {  
 String hash = password;  
 for (int x = 0; x < password.length(); x++) {  
 hash = *hash*(hash, password.charAt(x) + "");  
 }  
 return hash;  
 }  
  
 public static KeysetHandle generateKey() {  
 KeysetHandle keyHandle = null;  
 try {  
 AeadConfig.*register*();  
 keyHandle = KeysetHandle.*generateNew*(KeyTemplates.*get*("AES128\_GCM"));  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return keyHandle;  
 }  
  
 public static String keyToString(KeysetHandle keysetHandle) {  
 try {  
 ByteArrayOutputStream baos = new ByteArrayOutputStream();  
 ObjectOutputStream oos = new ObjectOutputStream(baos);  
 CleartextKeysetHandle.*write*(keysetHandle, JsonKeysetWriter.*withOutputStream*(oos));  
 oos.close();  
 return Arrays.*toString*(baos.toByteArray());  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public static byte[] ToByteArray(String array) {  
 array = array.replace("[", "").replace("]", "").replace(" ", "");  
 List<Byte> list = new ArrayList<>();  
 String[] splitArray = array.split(",");  
 for (String x : splitArray) {  
 list.add((byte) Integer.*parseInt*(x));  
 }  
 return Bytes.*toArray*(list);  
 }  
  
 public static KeysetHandle stringToKey(String keysetString) {  
 byte[] data = *ToByteArray*(keysetString);  
 System.*out*.println(data.length);  
 try {  
 ByteArrayInputStream bais = new ByteArrayInputStream(data);  
 ObjectInputStream ois = new ObjectInputStream(bais);  
 KeysetHandle keysetHandle = CleartextKeysetHandle.*read*(JsonKeysetReader.*withInputStream*(ois));  
 ois.close();  
 return keysetHandle;  
 } catch (IOException e) {  
 data[5] = (byte) -16;  
 return *stringToKey*(Arrays.*toString*(data));  
 } catch (GeneralSecurityException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public static byte[] decrypt(KeysetHandle keyHandle, byte[] cipher) {  
 try {  
 AeadConfig.*register*();  
 Aead aead = keyHandle.getPrimitive(Aead.class);  
 return aead.decrypt(cipher, ("password-manager").getBytes());  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public static byte[] encrypt(KeysetHandle keyHandle, byte[] data) {  
 try {  
 Aead aead = keyHandle.getPrimitive(Aead.class);  
 return aead.encrypt(data, ("password-manager").getBytes());  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
  
}

Manager/EnvelopeEncryption.java

package com.mycrolinks.passwdmgr.manager;  
  
import android.os.StrictMode;  
  
import java.io.IOException;  
import java.util.Objects;  
  
import okhttp3.FormBody;  
import okhttp3.OkHttpClient;  
import okhttp3.Request;  
import okhttp3.RequestBody;  
import okhttp3.Response;  
  
public class EnvelopeEncryption {  
 public final static String *domain* = "http://192.168.0.8:8080";  
  
 private static void threadOverride() {  
 StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();  
 StrictMode.*setThreadPolicy*(policy);  
 }  
  
 public static String encrypt(String text) {  
 *threadOverride*();  
 OkHttpClient client = new OkHttpClient();  
 RequestBody formBody = new FormBody.Builder().add("text", text).build();  
 Request request = new Request.Builder().url(*domain* + "/encrypt").post(formBody).build();  
 try {  
 Response response = client.newCall(request).execute();  
 return Objects.*requireNonNull*(response.body()).string();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public static String decrypt(String cipher) {  
 *threadOverride*();  
 OkHttpClient client = new OkHttpClient();  
 RequestBody formBody = new FormBody.Builder().add("cipher", cipher).build();  
 Request request = new Request.Builder().url(*domain* + "/decrypt").post(formBody).build();  
 try {  
 Response response = client.newCall(request).execute();  
 return Objects.*requireNonNull*(response.body()).string();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
}

Manager/MasterPassword.java

package com.mycrolinks.passwdmgr.manager;  
  
import android.os.StrictMode;  
  
import java.io.IOException;  
import java.util.Objects;  
  
import okhttp3.FormBody;  
import okhttp3.OkHttpClient;  
import okhttp3.Request;  
import okhttp3.RequestBody;  
import okhttp3.Response;  
  
public class MasterPassword {  
 private static void threadOverride() {  
 StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();  
 StrictMode.*setThreadPolicy*(policy);  
 }  
  
 public final static String *domain* = "http://192.168.0.8:8080";  
  
 public static boolean isUserExists(String email) {  
 *threadOverride*();  
 OkHttpClient client = new OkHttpClient();  
 RequestBody formBody = new FormBody.Builder().add("email", email).build();  
 Request request = new Request.Builder().url(*domain* + "/isUserExists").post(formBody).build();  
 try {  
 Response response = client.newCall(request).execute();  
 return Objects.*requireNonNull*(response.body()).string().equals("true");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
 public static boolean checkCredentials(String email, String password) {  
 *threadOverride*();  
 OkHttpClient client = new OkHttpClient();  
 RequestBody formBody = new FormBody.Builder().add("email", email).add("password", password).build();  
 Request request = new Request.Builder().url(*domain* + "/checkCredentials").post(formBody).build();  
 try {  
 Response response = client.newCall(request).execute();  
 return Objects.*requireNonNull*(response.body()).string().equals("true");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
 public static void setUser(String email, String password) {  
 *threadOverride*();  
 OkHttpClient client = new OkHttpClient();  
 RequestBody formBody = new FormBody.Builder().add("email", email).add("password", password).build();  
 Request request = new Request.Builder().url(*domain* + "/setUser").post(formBody).build();  
 try {  
 client.newCall(request).execute();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void delUser(String email) {  
 *threadOverride*();  
 OkHttpClient client = new OkHttpClient();  
 RequestBody formBody = new FormBody.Builder().add("email", email).build();  
 Request request = new Request.Builder().url(*domain* + "/delUser").post(formBody).build();  
 try {  
 client.newCall(request).execute();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

Manager/CustomAdapter.java

package com.mycrolinks.passwdmgr.manager;  
  
import android.annotation.SuppressLint;  
import android.content.ClipData;  
import android.content.ClipboardManager;  
import android.content.Context;  
import android.content.Intent;  
import android.database.DataSetObserver;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.EditText;  
import android.widget.ListAdapter;  
import android.widget.TextView;  
import android.widget.Toast;  
  
  
import androidx.appcompat.app.AlertDialog;  
  
import com.google.android.gms.auth.api.signin.GoogleSignIn;  
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;  
import com.mycrolinks.passwdmgr.R;  
import com.mycrolinks.passwdmgr.vault;  
  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
  
public class CustomAdapter implements ListAdapter {  
 PasswordItemList passwordItemList;  
 Context context;  
 GoogleSignInAccount signInAccount;  
 DateFormat df;  
  
 @SuppressLint("SimpleDateFormat")  
 public CustomAdapter(Context context, PasswordItemList passwordItemList) {  
 this.passwordItemList = passwordItemList;  
 this.context = context;  
 signInAccount = GoogleSignIn.*getLastSignedInAccount*(context);  
 df = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
 }  
  
 @Override  
 public boolean areAllItemsEnabled() {  
 return false;  
 }  
  
 @Override  
 public boolean isEnabled(int position) {  
 return true;  
 }  
  
 @Override  
 public void registerDataSetObserver(DataSetObserver observer) {  
 }  
  
 @Override  
 public void unregisterDataSetObserver(DataSetObserver observer) {  
 }  
  
 @Override  
 public int getCount() {  
 return passwordItemList.getSize();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return position;  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
  
 @Override  
 public boolean hasStableIds() {  
 return false;  
 }  
  
 @SuppressLint({"InflateParams", "SetTextI18n"})  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 PasswordItem pwdItem = passwordItemList.getPasswordItem(position);  
 if (convertView == null) {  
 LayoutInflater layoutInflater = LayoutInflater.*from*(context);  
 convertView = layoutInflater.inflate(R.layout.*list\_row*, null);  
 TextView title = convertView.findViewById(R.id.*list\_title*);  
 title.setText(pwdItem.getTitle());  
 convertView.setOnClickListener(v -> {  
 final View alertBox = layoutInflater.inflate(R.layout.*password\_item\_dialog*, null);  
 AlertDialog.Builder alertDialog = new AlertDialog.Builder(context);  
 alertDialog.setTitle("Credentials");  
 alertDialog.setView(alertBox);  
 EditText itemTitle = alertBox.findViewById(R.id.*alert\_title*);  
 itemTitle.setText(pwdItem.getTitle());  
 EditText itemPwd = alertBox.findViewById(R.id.*alert\_password*);  
 itemPwd.setText(pwdItem.getPassword());  
 TextView itemTime = alertBox.findViewById(R.id.*alert\_time*);  
 itemTime.setText("Last Modified : " + pwdItem.getTimestamp());  
 alertDialog.setIcon(R.drawable.*lock\_key*);  
 alertDialog.setPositiveButton("Save",  
 (dialog, which) -> {  
 Date today = Calendar.*getInstance*().getTime();  
 passwordItemList.update(position, new PasswordItem(itemTitle.getText().toString(), itemPwd.getText().toString(), df.format(today)));  
 Toast.*makeText*(context, "Credentials Saved", Toast.*LENGTH\_SHORT*).show();  
 Intent intent = new Intent(context, vault.class);  
 intent.putExtra("data", passwordItemList.getDocumentName());  
 context.startActivity(intent);  
 });  
 alertDialog.setNeutralButton("Copy",  
 (dialog, which) -> {  
 ClipboardManager clipboard = (ClipboardManager) context.getSystemService(Context.*CLIPBOARD\_SERVICE*);  
 ClipData clip = ClipData.*newPlainText*(pwdItem.getTitle(), pwdItem.getPassword());  
 clipboard.setPrimaryClip(clip);  
 dialog.cancel();  
 Toast.*makeText*(context, "Password Copied to Clipboard", Toast.*LENGTH\_SHORT*).show();  
 });  
 alertDialog.setNegativeButton("Delete",  
 (dialog, which) -> {  
 passwordItemList.delete(position);  
 Toast.*makeText*(context, "Credentials Deleted", Toast.*LENGTH\_SHORT*).show();  
 Intent intent = new Intent(context, vault.class);  
 intent.putExtra("data", passwordItemList.getDocumentName());  
 context.startActivity(intent);  
 });  
 alertDialog.show();  
 });  
  
 }  
 return convertView;  
 }  
  
 @Override  
 public int getItemViewType(int position) {  
 return position;  
 }  
  
 @Override  
 public int getViewTypeCount() {  
 return passwordItemList.getSize();  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return false;  
 }  
}

Main\_Activity.java

package com.mycrolinks.passwdmgr;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.gms.auth.api.signin.GoogleSignIn;  
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;  
import com.google.android.gms.auth.api.signin.GoogleSignInClient;  
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;  
import com.google.android.gms.common.api.ApiException;  
import com.google.android.gms.tasks.Task;  
import com.google.firebase.auth.AuthCredential;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.auth.GoogleAuthProvider;  
import com.mycrolinks.passwdmgr.manager.MasterPassword;  
  
import java.util.Objects;  
  
public class MainActivity extends AppCompatActivity {  
 GoogleSignInClient mGoogleSignInClient;  
 private static final int *RC\_SIGN\_IN* = 1000;  
 private FirebaseAuth mAuth;  
 FirebaseUser currentUser;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 mAuth = FirebaseAuth.*getInstance*();  
 // Configure Google Sign In  
 GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.*DEFAULT\_SIGN\_IN*)  
 .requestIdToken(getString(R.string.*default\_web\_client\_id*))  
 .requestEmail()  
 .build();  
  
 mGoogleSignInClient = GoogleSignIn.*getClient*(this, gso);  
 }  
  
 @SuppressWarnings("deprecation")  
 public void sign(View v) {  
 Toast.*makeText*(this, "Signing you in", Toast.*LENGTH\_SHORT*).show();  
 Intent signInIntent = mGoogleSignInClient.getSignInIntent();  
 startActivityForResult(signInIntent, *RC\_SIGN\_IN*);  
 }  
  
 @Override  
 public void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == *RC\_SIGN\_IN*) {  
 Task<GoogleSignInAccount> task = GoogleSignIn.*getSignedInAccountFromIntent*(data);  
 try {  
 GoogleSignInAccount account = task.getResult(ApiException.class);  
 firebaseAuthWithGoogle(account.getIdToken());  
 } catch (ApiException e) {  
 Toast.*makeText*(this, e.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
// @Override  
// public void onStart() {  
// super.onStart();  
// currentUser = mAuth.getCurrentUser();  
// if (currentUser != null) {  
// vaultRedirect();  
// }  
// }  
  
 private void firebaseAuthWithGoogle(String idToken) {  
 AuthCredential credential = GoogleAuthProvider.*getCredential*(idToken, null);  
 mAuth.signInWithCredential(credential)  
 .addOnCompleteListener(this, task -> {  
 if (task.isSuccessful()) {  
 System.*out*.println("Logged in");  
 currentUser = mAuth.getCurrentUser();  
 vaultRedirect();  
 } else {  
 Toast.*makeText*(MainActivity.this, "Sorry, Authentication Failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 private void vaultRedirect() {  
 if (MasterPassword.*isUserExists*(Objects.*requireNonNull*(currentUser.getEmail()))) {  
 startActivity(new Intent(getApplicationContext(), vault\_pin.class));  
 } else {  
 startActivity(new Intent(getApplicationContext(), create\_vault.class));  
 }  
 }  
}

Create\_vault.java

package com.mycrolinks.passwdmgr;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.gms.auth.api.signin.GoogleSignIn;  
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;  
import com.google.crypto.tink.KeysetHandle;  
import com.google.firebase.firestore.FirebaseFirestore;  
import com.mycrolinks.passwdmgr.manager.Encryptor;  
import com.mycrolinks.passwdmgr.manager.EnvelopeEncryption;  
import com.mycrolinks.passwdmgr.manager.MasterPassword;  
import com.mycrolinks.passwdmgr.manager.PasswordItemList;  
  
import java.util.Arrays;  
import java.util.HashMap;  
import java.util.Map;  
  
public class create\_vault extends AppCompatActivity {  
 EditText pass\_new, pass\_conf;  
 Button create;  
  
 GoogleSignInAccount signInAccount;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_create\_vault*);  
 pass\_new = findViewById(R.id.*pass\_new*);  
 pass\_conf = findViewById(R.id.*pass\_conf*);  
 create = findViewById(R.id.*create*);  
 create.setOnClickListener(view -> {  
 if (pass\_new.getText().toString().equals(pass\_conf.getText().toString())) {  
 signInAccount = GoogleSignIn.*getLastSignedInAccount*(this);  
 assert signInAccount != null;  
 String email = signInAccount.getEmail();  
 String rawPassword = pass\_conf.getText().toString();  
 String hashPassword = Encryptor.*hash\_wrapper*(rawPassword);  
 String documentName = Encryptor.*hash\_wrapper*(email + hashPassword);  
 Map<String, Object> document = new HashMap<>();  
 PasswordItemList passwordItemList = new PasswordItemList();  
 KeysetHandle keyHandle = Encryptor.*generateKey*();  
 byte[] passwordListByteArray = passwordItemList.toByteArray();  
 byte[] encryptedData = Encryptor.*encrypt*(keyHandle, passwordListByteArray);  
 String keyString = Encryptor.*keyToString*(keyHandle);  
 String encryptedKey = EnvelopeEncryption.*encrypt*(keyString);  
 System.*out*.println("keyhandle text : " + keyString);  
 System.*out*.println("cipher text : " + encryptedKey);  
 document.put("key", encryptedKey);  
 document.put("data", Arrays.*toString*(encryptedData));  
 document.put("author", email);  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
 db.collection("root").document(documentName).set(document)  
 .addOnSuccessListener(aVoid -> {  
 Toast.*makeText*(this, "New Vault Created", Toast.*LENGTH\_SHORT*).show();  
 MasterPassword.*setUser*(email, hashPassword);  
 Toast.*makeText*(this, "Vault created successfully", Toast.*LENGTH\_SHORT*).show();  
 startActivity(new Intent(this, vault\_pin.class));  
 }  
 )  
 .addOnFailureListener(e -> {  
 System.*out*.println("Error writing document" + e.getMessage());  
 startActivity(new Intent(this, create\_vault.class));  
 });  
 } else {  
 Toast.*makeText*(this, "Passwords Didn't Match", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
}

vault\_pin.java

package com.mycrolinks.passwdmgr;  
  
import android.annotation.SuppressLint;  
import android.content.Intent;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.CheckBox;  
import android.widget.EditText;  
import android.widget.Toast;  
  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.gms.auth.api.signin.GoogleSignIn;  
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;  
import com.google.android.gms.common.api.GoogleApiClient;  
import com.google.android.gms.common.api.Status;  
import com.google.android.gms.safetynet.SafetyNet;  
import com.google.firebase.auth.FirebaseAuth;  
import com.mycrolinks.passwdmgr.manager.Encryptor;  
import com.mycrolinks.passwdmgr.manager.MasterPassword;  
  
@SuppressWarnings("deprecation")  
public class vault\_pin extends AppCompatActivity implements GoogleApiClient.ConnectionCallbacks {  
 GoogleSignInAccount signInAccount;  
 Button unlock, logout, delete;  
 CheckBox checkbox;  
 GoogleApiClient googleApiClient;  
 String SITE\_KEY = "6LcbWZgbAAAAAGj9z9VVweX8s5htypF-5ykAzM5V";  
  
 @SuppressLint("MissingPermission")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_vault\_pin*);  
 unlock = findViewById(R.id.*signin*);  
 logout = findViewById(R.id.*log\_but*);  
 delete = findViewById(R.id.*del\_but*);  
 checkbox = findViewById(R.id.*checkbox*);  
 signInAccount = GoogleSignIn.*getLastSignedInAccount*(this);  
 googleApiClient = new GoogleApiClient.Builder(this).addApi(SafetyNet.*API*).addConnectionCallbacks(vault\_pin.this).build();  
 googleApiClient.connect();  
 //checkbox.setSelected(true);  
 checkbox.setOnClickListener(v -> {  
 if (checkbox.isChecked()) {  
 SafetyNet.*SafetyNetApi*.verifyWithRecaptcha(googleApiClient, SITE\_KEY).setResultCallback(recaptchaTokenResult -> {  
 Status status = recaptchaTokenResult.getStatus();  
 if (status.isSuccess()) {  
 Toast.*makeText*(vault\_pin.this, "Captcha Verified Successfully", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 checkbox.setSelected(false);  
 }  
 });  
 }  
 });  
  
 logout.setOnClickListener(view -> {  
 FirebaseAuth.*getInstance*().signOut();  
 Intent intent = new Intent(getApplicationContext(), MainActivity.class);  
 startActivity(intent);  
 });  
 unlock.setOnClickListener(view -> {  
 EditText passwordEditText = findViewById(R.id.*password\_editText*);  
 String hashPassword = Encryptor.*hash\_wrapper*(passwordEditText.getText().toString());  
  
 if(checkbox.isChecked()){  
 String email = signInAccount.getEmail();  
 if (MasterPassword.*checkCredentials*(email,hashPassword)) {  
 Intent intent = new Intent(getApplicationContext(), vault.class);  
 String documentName = Encryptor.*hash\_wrapper*(email + hashPassword);  
 intent.putExtra("data", documentName);  
 startActivity(intent);  
 } else {  
 Toast.*makeText*(this, "Wrong Password", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 Toast.*makeText*(this, "Verify ReCAPTCHA", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
  
 }  
  
  
 @Override  
 public void onConnected(@Nullable @org.jetbrains.annotations.Nullable Bundle bundle) {  
  
 }  
  
 @Override  
 public void onConnectionSuspended(int i) {  
  
 }  
}

vault.java

package com.mycrolinks.passwdmgr;  
  
import android.annotation.SuppressLint;  
import android.content.ClipData;  
import android.content.ClipboardManager;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ListView;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.gms.auth.api.signin.GoogleSignIn;  
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;  
import com.google.crypto.tink.KeysetHandle;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.firestore.DocumentReference;  
import com.google.firebase.firestore.DocumentSnapshot;  
import com.google.firebase.firestore.FirebaseFirestore;  
import com.mycrolinks.passwdmgr.manager.CustomAdapter;  
import com.mycrolinks.passwdmgr.manager.Encryptor;  
import com.mycrolinks.passwdmgr.manager.EnvelopeEncryption;  
import com.mycrolinks.passwdmgr.manager.MasterPassword;  
import com.mycrolinks.passwdmgr.manager.PasswordItem;  
import com.mycrolinks.passwdmgr.manager.PasswordItemList;  
  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.Objects;  
  
public class vault extends AppCompatActivity {  
  
 GoogleSignInAccount signInAccount;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_vault*);  
 signInAccount = GoogleSignIn.*getLastSignedInAccount*(this);  
 final ListView list = findViewById(R.id.*list*);  
 String documentName = getIntent().getStringExtra("data");  
 DocumentReference documentReference = db.collection("root").document(documentName);  
 documentReference.get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 DocumentSnapshot document = task.getResult();  
 if (document.exists()) {  
 String encryptedPasswordList = (String) Objects.*requireNonNull*(document.getData()).get("data");  
 String encryptedKeysetHandle = (String) Objects.*requireNonNull*(document.getData()).get("key");  
 System.*out*.println("encrypted" + encryptedKeysetHandle);  
 String decrypted = EnvelopeEncryption.*decrypt*(encryptedKeysetHandle);  
 System.*out*.println("decrypted" + decrypted);  
 KeysetHandle keysetHandle = Encryptor.*stringToKey*(decrypted);  
 PasswordItemList passwordItemList = new PasswordItemList(Encryptor.*decrypt*(keysetHandle, Encryptor.*ToByteArray*(encryptedPasswordList)));  
 passwordItemList.setDocumentName(documentName);  
 passwordItemList.setKeysetHandle(keysetHandle);  
 CustomAdapter customAdapter = new CustomAdapter(this, passwordItemList);  
 list.setAdapter(customAdapter);  
 Button exit = findViewById(R.id.*exit*);  
 exit.setOnClickListener(v -> {  
 FirebaseAuth.*getInstance*().signOut();  
 Intent intent = new Intent(getApplicationContext(), MainActivity.class);  
 startActivity(intent);  
 });  
 Button add = findViewById(R.id.*add*);  
 add.setOnClickListener(v -> {  
 LayoutInflater layoutInflater = LayoutInflater.*from*(this);  
 final View alertBox = layoutInflater.inflate(R.layout.*password\_item\_dialog*, null);  
 AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);  
 alertDialog.setTitle("Credentials");  
 alertDialog.setView(alertBox);  
 EditText itemTitle = alertBox.findViewById(R.id.*alert\_title*);  
 EditText itemPwd = alertBox.findViewById(R.id.*alert\_password*);  
 alertDialog.setIcon(R.drawable.*lock\_key*);  
 alertDialog.setPositiveButton("Save",  
 (dialog, which) -> {  
 Date today = Calendar.*getInstance*().getTime();  
 @SuppressLint("SimpleDateFormat") DateFormat df = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
 passwordItemList.add(new PasswordItem(itemTitle.getText().toString(), itemPwd.getText().toString(), df.format(today)));  
 Toast.*makeText*(this, "Credentials Saved", Toast.*LENGTH\_SHORT*).show();  
 Intent intent = new Intent(this, vault.class);  
 intent.putExtra("data", documentName);  
 this.startActivity(intent);  
 });  
 alertDialog.setNeutralButton("Copy",  
 (dialog, which) -> {  
 ClipboardManager clipboard = (ClipboardManager) this.getSystemService(Context.*CLIPBOARD\_SERVICE*);  
 ClipData clip = ClipData.*newPlainText*(itemTitle.getText().toString(), itemPwd.getText().toString());  
 clipboard.setPrimaryClip(clip);  
 dialog.cancel();  
 Toast.*makeText*(this, "Password Copied to Clipboard", Toast.*LENGTH\_SHORT*).show();  
 });  
 alertDialog.show();  
  
 });  
 Button delete = findViewById(R.id.*del\_but*);  
 delete.setOnClickListener(view -> documentReference.delete()  
 .addOnSuccessListener(aVoid -> {  
 MasterPassword.*delUser*(signInAccount.getEmail());  
 Toast.*makeText*(vault.this, "Vault successfully deleted!", Toast.*LENGTH\_SHORT*).show();  
 exit.performClick();  
 })  
 .addOnFailureListener(e -> {  
 Toast.*makeText*(vault.this, "Error deleting vault", Toast.*LENGTH\_SHORT*).show();  
 System.*out*.println(e.getMessage());  
 }));  
 } else {  
 System.*out*.println("No such document");  
 }  
 } else {  
 System.*out*.println("get failed with " + task.getException());  
 }  
 });  
  
  
 }  
}

activity\_create\_vault.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".create\_vault"  
 android:layout\_margin="10dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="right"  
 android:orientation="horizontal">  
 <Button  
 android:id="@+id/create"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Create Vault"  
 android:layout\_margin="10dp"/>  
 </LinearLayout>  
 <Space  
 android:layout\_width="wrap\_content"  
 android:layout\_height="20dp" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:text="Create Vault"  
 android:textSize="20sp" />  
 <Space  
 android:layout\_width="wrap\_content"  
 android:layout\_height="20dp" />  
  
 <ImageView  
 android:layout\_width="150dp"  
 android:layout\_height="150dp"  
 android:layout\_gravity="center"  
 android:src="@drawable/lock\_key" />  
  
 <Space  
 android:layout\_width="wrap\_content"  
 android:layout\_height="20dp" />  
 <EditText  
 android:id="@+id/pass\_new"  
 android:layout\_width="325dp"  
 android:layout\_height="wrap\_content"  
 android:hint="New Password"  
 android:layout\_gravity="center"  
 android:inputType="textPassword"  
 />  
 <Space  
 android:layout\_width="wrap\_content"  
 android:layout\_height="20dp" />  
 <EditText  
 android:id="@+id/pass\_conf"  
 android:layout\_width="325dp"  
 android:layout\_height="wrap\_content"  
 android:hint="Confirm Password"  
 android:layout\_gravity="center"  
 android:inputType="textPassword" />  
</LinearLayout>

Activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="514dp"  
 android:gravity="center"  
 android:onClick="sign"  
 android:background="@color/colorPrimary"  
 android:text="Sign In with Google"  
 />  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="150dp"  
 android:layout\_height="150dp"  
 android:layout\_centerHorizontal="true"  
 android:layout\_centerVertical="true"  
 app:srcCompat="@drawable/fui\_ic\_googleg\_color\_24dp" />  
  
</RelativeLayout>

Activity\_vault.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:paddingLeft="16dp"  
 android:paddingTop="16dp"  
 android:paddingRight="16dp"  
 android:paddingBottom="16dp"  
 tools:context=".vault">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="right"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/del\_but"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:layout\_gravity="right"  
 android:text="Delete Vault" />  
  
 <Button  
 android:id="@+id/add"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="right"  
 android:layout\_margin="10dp"  
 android:text="Add" />  
  
 <Button  
 android:id="@+id/exit"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="right"  
 android:layout\_margin="10dp"  
 android:text="Exit" />  
  
 </LinearLayout>  
  
 <ListView  
 android:id="@+id/list"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="600dp"  
 android:divider="#000"  
 android:dividerHeight="1dp"  
 android:footerDividersEnabled="false"  
 android:headerDividersEnabled="false"  
 android:isScrollContainer="true"  
 android:scrollbarAlwaysDrawVerticalTrack="true" />  
  
  
</LinearLayout>

Activity\_vault\_pin.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".vault\_pin">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:gravity="right"  
 android:orientation="horizontal">  
  
 <CheckBox  
 android:id="@+id/checkbox"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:text="I'm not a robot" />  
  
 <Button  
 android:id="@+id/log\_but"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:text="Logout" />  
  
 <Button  
 android:id="@+id/signin"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:text="Unlock" />  
 </LinearLayout>  
  
 <Space  
 android:layout\_width="wrap\_content"  
 android:layout\_height="20dp" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:text="Master Password"  
 android:textSize="20sp" />  
  
 <Space  
 android:layout\_width="wrap\_content"  
 android:layout\_height="20dp" />  
  
 <ImageView  
 android:layout\_width="150dp"  
 android:layout\_height="150dp"  
 android:layout\_gravity="center"  
 android:src="@drawable/lock\_key" />  
  
 <Space  
 android:layout\_width="wrap\_content"  
 android:layout\_height="20dp" />  
  
 <EditText  
 android:id="@+id/password\_editText"  
 android:layout\_width="325dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:hint="Enter Master Password"  
 android:inputType="textPassword" />  
  
</LinearLayout>

List\_row.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="5dip">  
 <TextView  
 android:id="@+id/list\_title"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:textSize="25sp"  
 android:gravity="center"  
 />  
</LinearLayout>

Password\_item\_dialog.xml

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
 <EditText android:inputType="textAutoComplete"  
 android:textSize="20sp"  
 android:id="@+id/alert\_title"  
 android:scrollHorizontally="true"  
 android:layout\_height="70dp"  
 android:hint="Title / Username"  
 android:layout\_width="250dp"  
 android:layout\_gravity="center"  
 android:autofillHints="username" />  
 <EditText android:inputType="textVisiblePassword"  
 android:textSize="20sp"  
 android:id="@+id/alert\_password"  
 android:scrollHorizontally="true"  
 android:layout\_height="70dp"  
 android:hint="Password"  
 android:layout\_width="250dp"  
 android:layout\_gravity="center"  
 android:autofillHints="password" />  
 <TextView  
 android:textSize="20sp"  
 android:id="@+id/alert\_time"  
 android:scrollHorizontally="true"  
 android:layout\_height="70dp"  
 android:layout\_width="250dp"  
 android:layout\_gravity="center"  
 android:autofillHints="password" />  
  
</LinearLayout>

Manifest.xml

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.mycrolinks.passwdmgr">  
 <uses-permission android:name="android.permission.INTERNET" />  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:networkSecurityConfig="@xml/network\_security\_config"  
 android:theme="@style/Theme.PasswordManager">  
 <activity android:name=".vault" />  
 <activity android:name=".create\_vault" />  
 <activity android:name=".vault\_pin" />  
 <activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
</manifest>

**NodeJS REST API:**

App.js

const dotenv = require('dotenv');  
dotenv.config();  
const asymmetricEncryption = require("./asymmetricEncryption")  
const ***express*** = require('express');  
const cors = require('cors');  
const fs = require("fs");  
const app = ***express***();  
const port = 8080  
  
// noinspection JSCheckFunctionSignatures  
app.use(cors({  
 origin: '\*'  
}));  
  
app.use(***express***.***urlencoded***({  
 extended: true  
}))  
  
function parseKeySet(text) {  
 let byteArray = text.toString().replace(/]/, "").replace(/\[/, "").split(",");  
 ***console***.log(byteArray.length)  
 byteArray.splice(0, 6);  
 ***console***.log(***JSON***.stringify(byteArray))  
  
 let jsonString = ""  
 byteArray.forEach(element => jsonString += ***String***.fromCharCode(parseInt(element.trim())))  
 ***console***.log(jsonString)  
 let json = ***JSON***.parse(jsonString)  
 return {primaryKeyId: json.primaryKeyId, value: json.key[0].keyData.value}  
}  
  
function stringToByteArray(string) {  
 let myBuffer = [-84, -19, 0, 5, 119, -18];  
 const buffer = ***Buffer***.from(string, 'utf8');  
 for (let i = 0; i < buffer.length; i++) {  
 myBuffer.push(buffer[i]);  
 }  
 myBuffer.push(10)  
 return myBuffer  
}  
  
function constructKeyset(keysetValues) {  
 keysetValues = ***JSON***.parse(keysetValues)  
 let keyset = {  
 "primaryKeyId": keysetValues.primaryKeyId,  
 "key": [{  
 "keyData": {  
 "typeUrl": "type.googleapis.com/google.crypto.tink.AesGcmKey",  
 "value": keysetValues.value,  
 "keyMaterialType": "SYMMETRIC"  
 }, "status": "ENABLED", "keyId": keysetValues.primaryKeyId, "outputPrefixType": "TINK"  
 }]  
 }  
  
 let jsonString = ***JSON***.stringify(keyset)  
 ***console***.log(jsonString)  
 return stringToByteArray(jsonString)  
}  
  
app.post('/encrypt', function (req, res) {  
 res.setHeader('Content-Type', 'application/json');  
 let keysetValues = parseKeySet(req.body.text)  
 ***console***.log(keysetValues)  
 asymmetricEncryption.encrypt(***Buffer***.from(***JSON***.stringify(keysetValues), 'utf8'))  
 .then(cipher => {  
 ***console***.log("Encryption : " + cipher)  
 res.end(***JSON***.stringify({cipher}))  
 })  
 .catch(err => {  
 ***console***.log("There was an error", err)  
 })  
});  
app.post('/decrypt', function (req, res) {  
 ***console***.log("decrypt", req.body)  
 res.setHeader('Content-Type', 'application/json');  
 let byteArray = ***JSON***.parse(req.body.cipher).cipher.data  
 ***console***.log(byteArray.length)  
 asymmetricEncryption.decrpyt(***Buffer***.from(byteArray)).then(keysetValues => {  
 let keyset = constructKeyset(keysetValues)  
 ***console***.log(keyset.length)  
 ***console***.log("Decryption : " + ***JSON***.stringify(keyset))  
 return res.end(***JSON***.stringify(keyset))  
 }).catch(err => {  
 ***console***.log("There was an error", err)  
 })  
});  
  
app.post('/isUserExists', function (req, res) {  
 ***console***.log("isUserExists", req.body)  
 res.setHeader('Content-Type', 'application/json');  
 const data = require("./masterPassword.json")  
 ***console***.log(data[req.body.email], data[req.body.email] !== undefined);  
 return res.end((data[req.body.email] !== undefined).toString())  
});  
  
app.post('/checkCredentials', function (req, res) {  
 ***console***.log("checkCredentials", req.body)  
 res.setHeader('Content-Type', 'application/json');  
 const data = require("./masterPassword.json")  
 return res.end((data[req.body.email] === req.body.password).toString())  
});  
  
app.post('/setUser', function (req, res) {  
 ***console***.log("setUser", req.body)  
 res.setHeader('Content-Type', 'application/json');  
 const data = require("./masterPassword.json")  
 data[req.body.email] = req.body.password  
 fs.writeFile("masterPassword.json", ***JSON***.stringify(data), err => {  
 if (err) {  
 ***console***.log(err)  
 return res.end(***JSON***.stringify({result: false}))  
 } else {  
 return res.end(***JSON***.stringify({result: true}))  
 }  
 });  
});  
  
app.post('/delUser', function (req, res) {  
 ***console***.log("delUser", req.body)  
 res.setHeader('Content-Type', 'application/json');  
 const data = require("./masterPassword.json")  
 delete data[req.body.email];  
 fs.writeFile("masterPassword.json", ***JSON***.stringify(data), err => {  
 if (err) {  
 ***console***.log(err)  
 return res.end(***JSON***.stringify({result: false}))  
 } else {  
 return res.end(***JSON***.stringify({result: true}))  
 }  
 });  
});  
  
app.listen(port, () => {  
 ***console***.log(`Password-Manager Server listening at http://localhost:${port}`)  
})

asymmetricEncryption.js

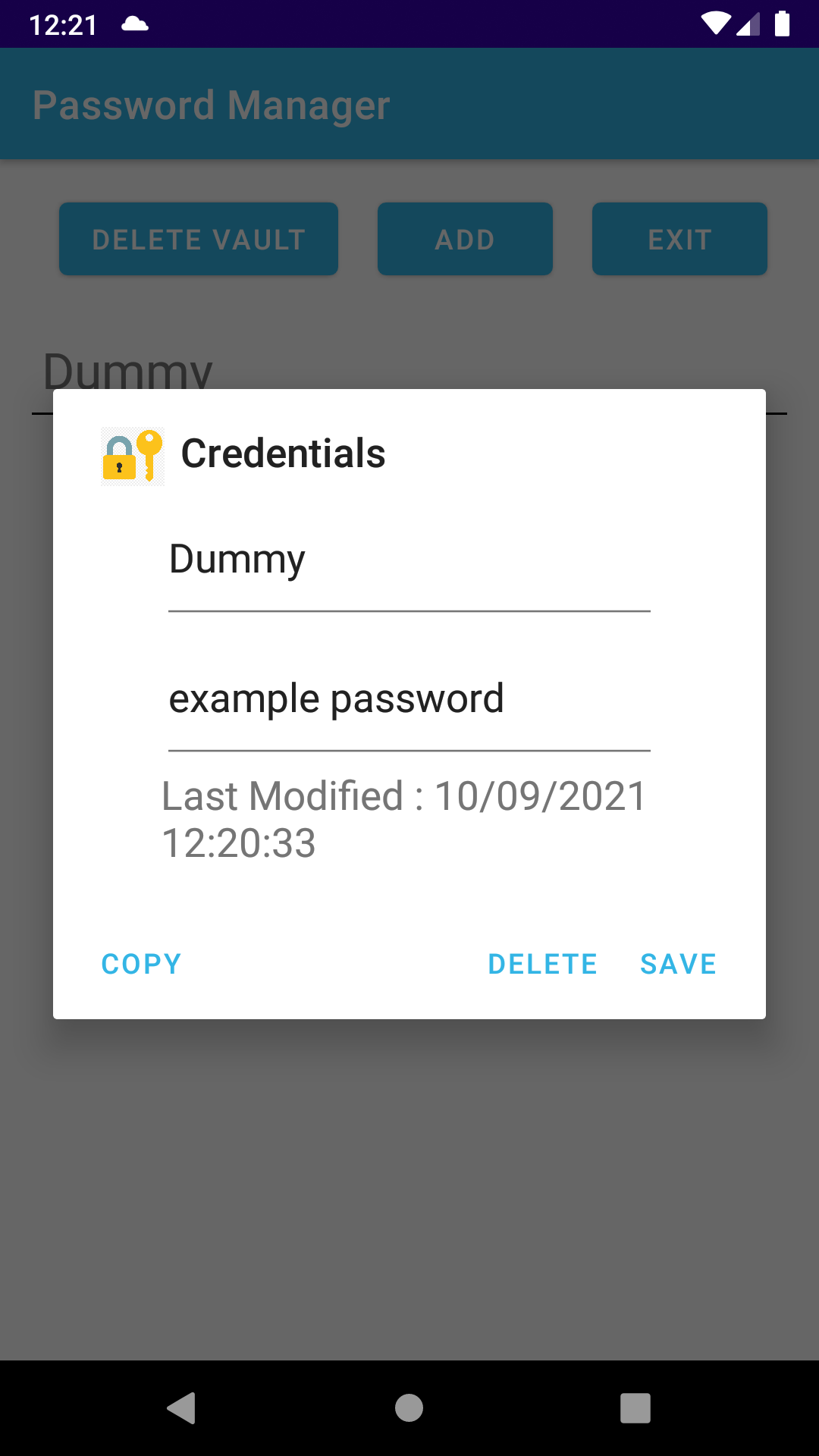
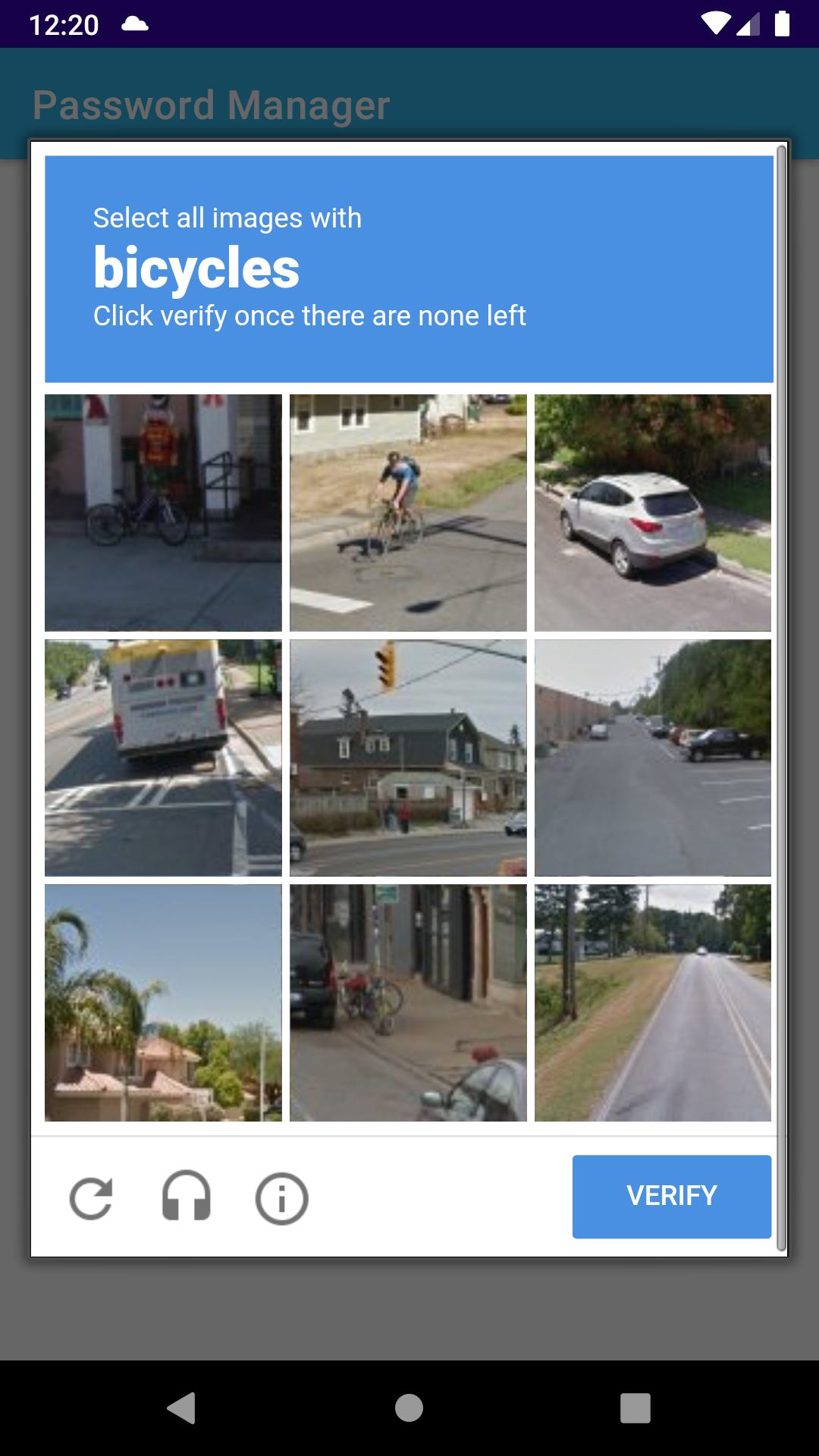
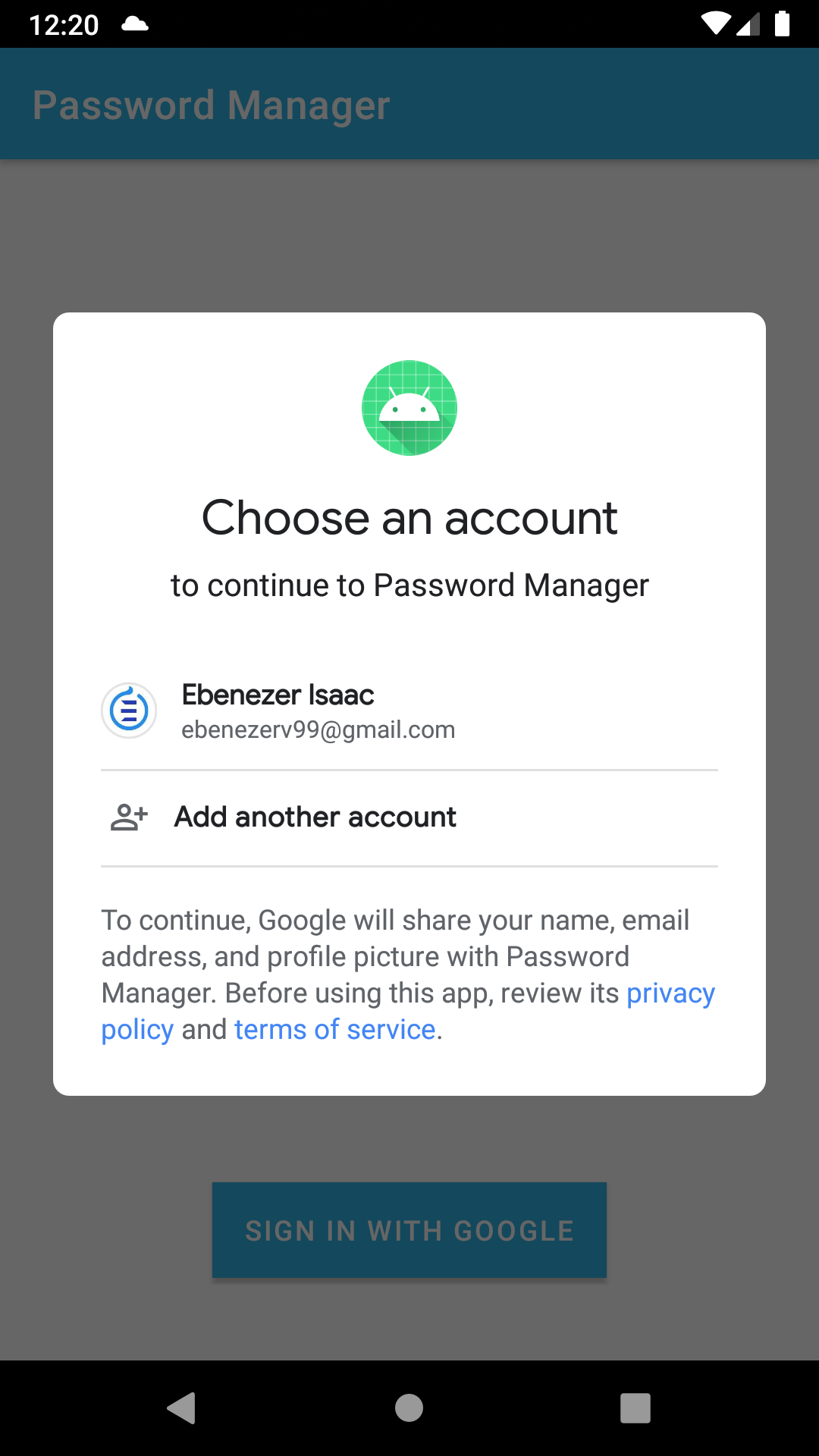
const projectId = 'premium-bloom-319904';  
const locationId = 'global';  
const keyRingId = 'password-manager';  
const keyId = 'password-manager';  
const versionId = '1';  
const {KeyManagementServiceClient} = require('@google-cloud/kms');  
const client = new KeyManagementServiceClient();  
const versionName = client.cryptoKeyVersionPath(  
 projectId,  
 locationId,  
 keyRingId,  
 keyId,  
 versionId  
);  
const ***crc32c*** = require('fast-crc32c');  
  
async function decrypt(ciphertext) {  
  
 const ciphertextCrc32c = ***crc32c***.calculate(ciphertext);  
  
 async function decryptAsymmetric() {  
 const [decryptResponse] = await client.asymmetricDecrypt({  
 name: versionName,  
 ciphertext: ciphertext,  
 ciphertextCrc32c: {  
 value: ciphertextCrc32c,  
 },  
 });  
 if (!decryptResponse.verifiedCiphertextCrc32c) {  
 throw new ***Error***('AsymmetricDecrypt: request corrupted in-transit');  
 }  
 if (  
 ***crc32c***.calculate(decryptResponse.plaintext) !==  
 ***Number***(decryptResponse.plaintextCrc32c.value)  
 ) {  
 throw new Error('AsymmetricDecrypt: response corrupted in-transit');  
 }  
 return decryptResponse.plaintext.toString();  
 }  
  
 return decryptAsymmetric();  
}  
  
async function encrypt(plaintextBuffer) {  
 async function encryptAsymmetric() {  
 console.log(0)  
 const [publicKey] = await client.getPublicKey({  
 name: versionName,  
 });  
 console.log(1)  
 if (publicKey.name !== versionName) {  
 throw new Error('GetPublicKey: request corrupted in-transit');  
 }  
 if (crc32c.calculate(publicKey.pem) !== Number(publicKey.pemCrc32c.value)) {  
 throw new Error('GetPublicKey: response corrupted in-transit');  
 }  
 const crypto = require('crypto');  
 return crypto.publicEncrypt(  
 {  
 key: publicKey.pem,  
 oaepHash: 'sha256'  
 },  
 plaintextBuffer  
 );  
 }  
  
 return encryptAsymmetric();  
}  
  
module.exports.encrypt = encrypt;  
module.exports.decrpyt = decrypt;

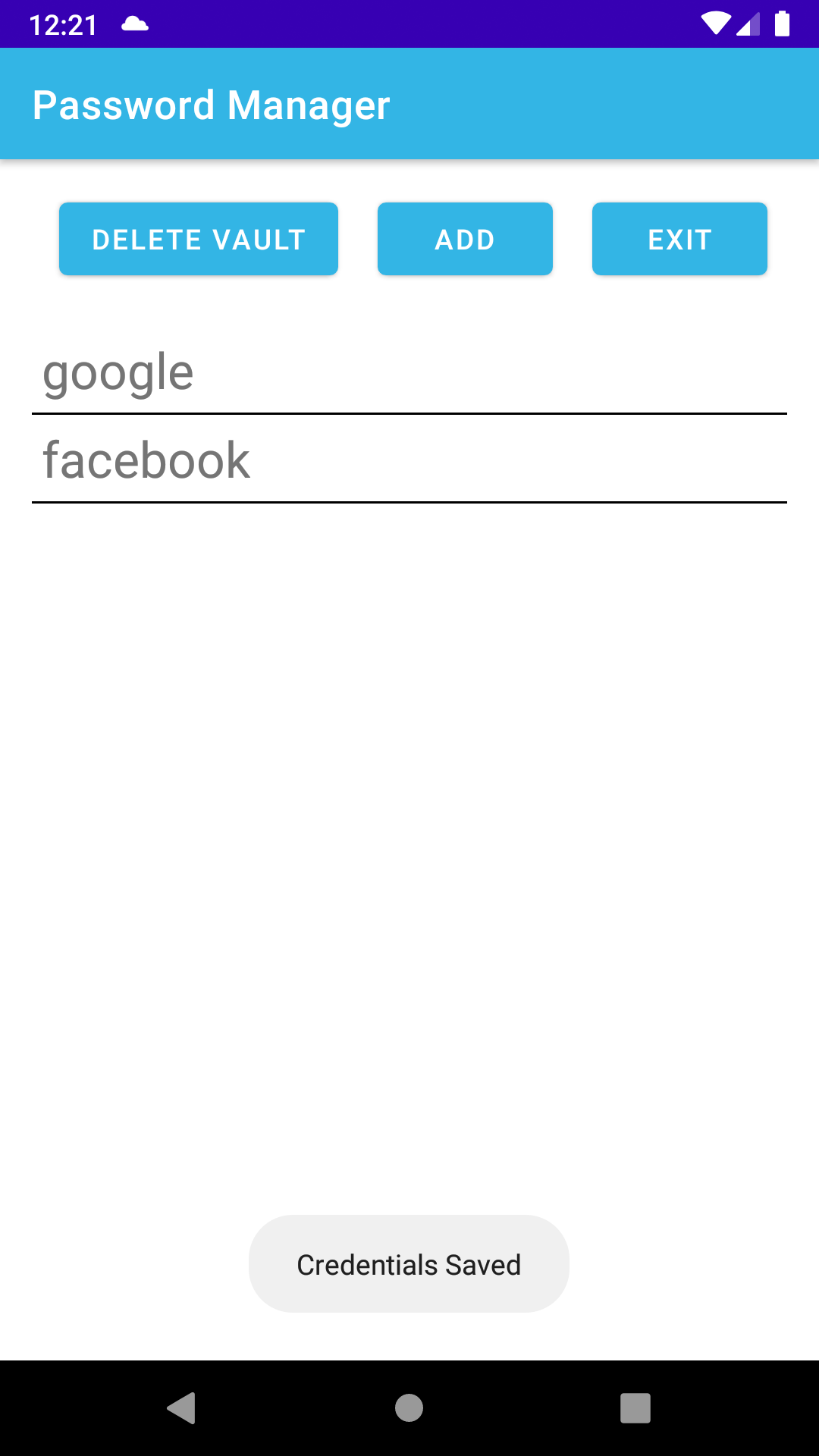
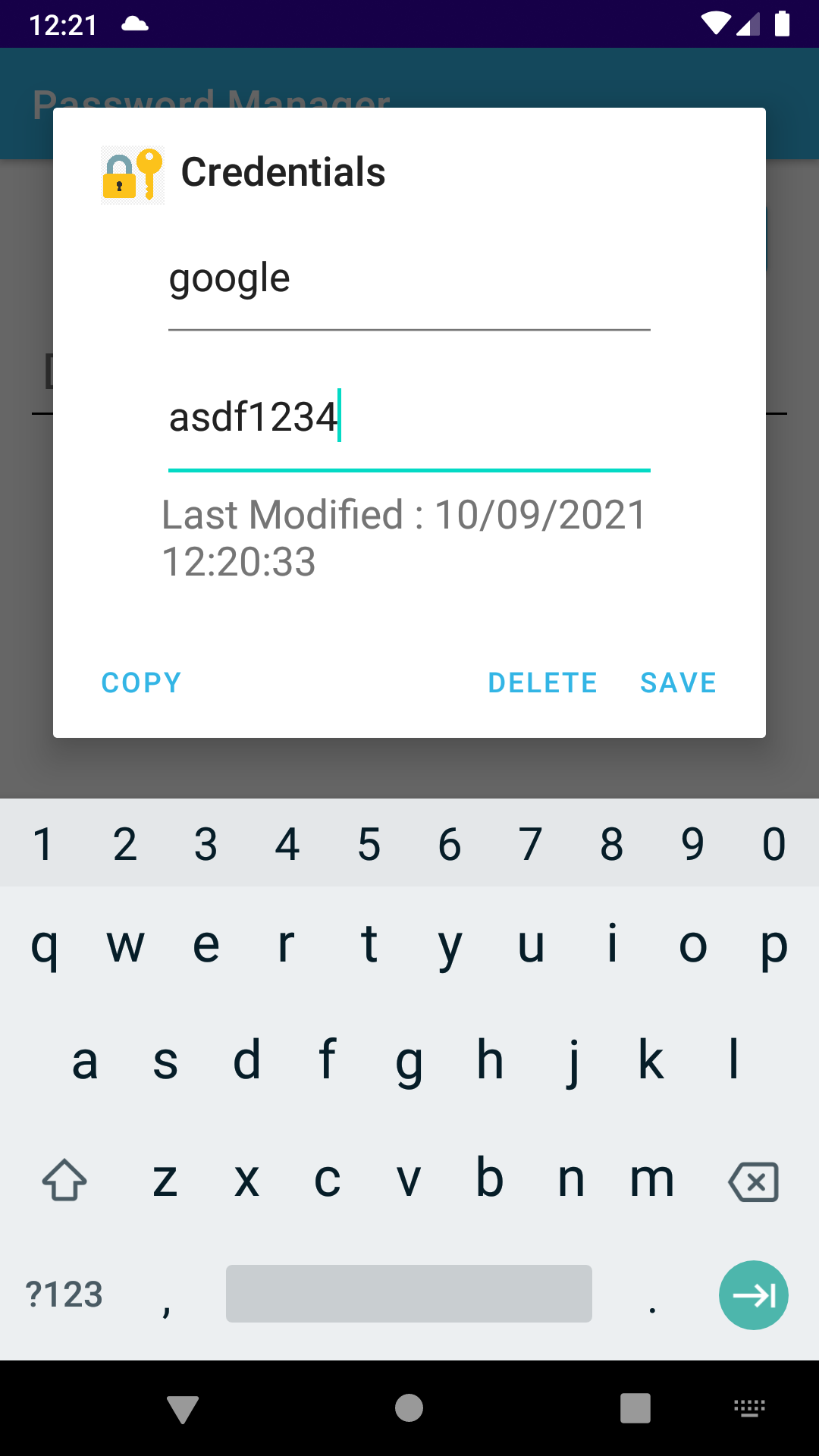
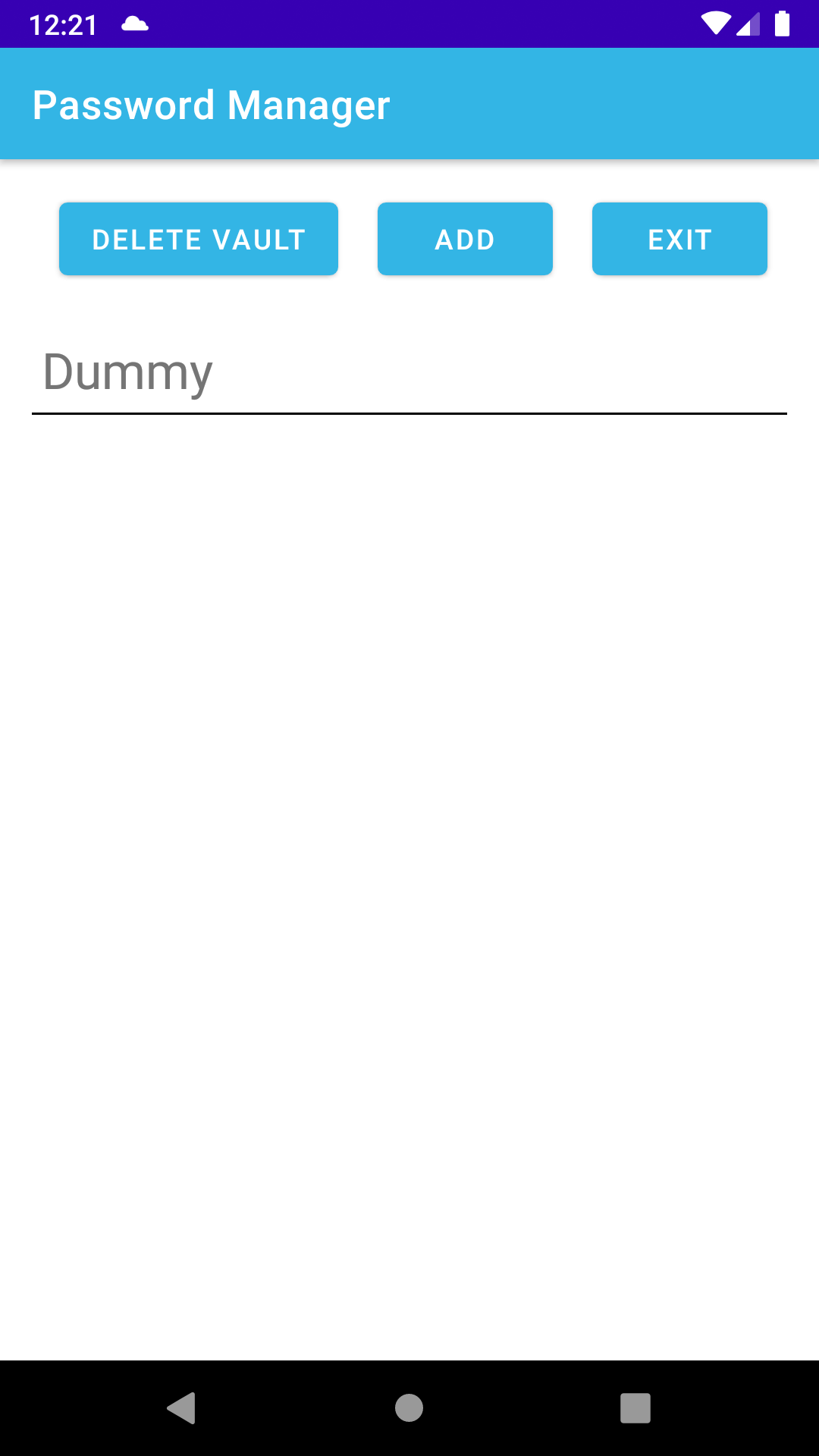
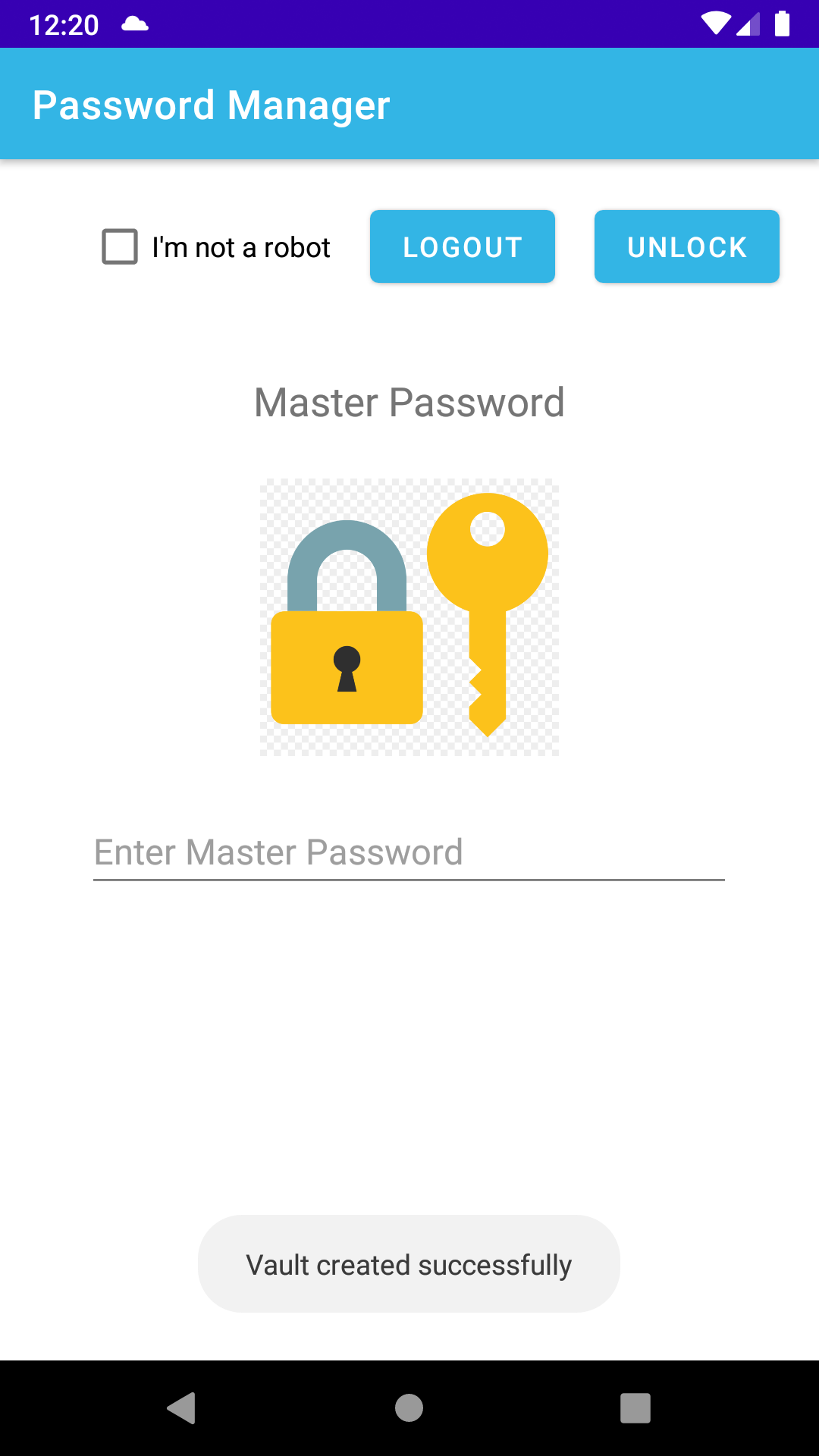
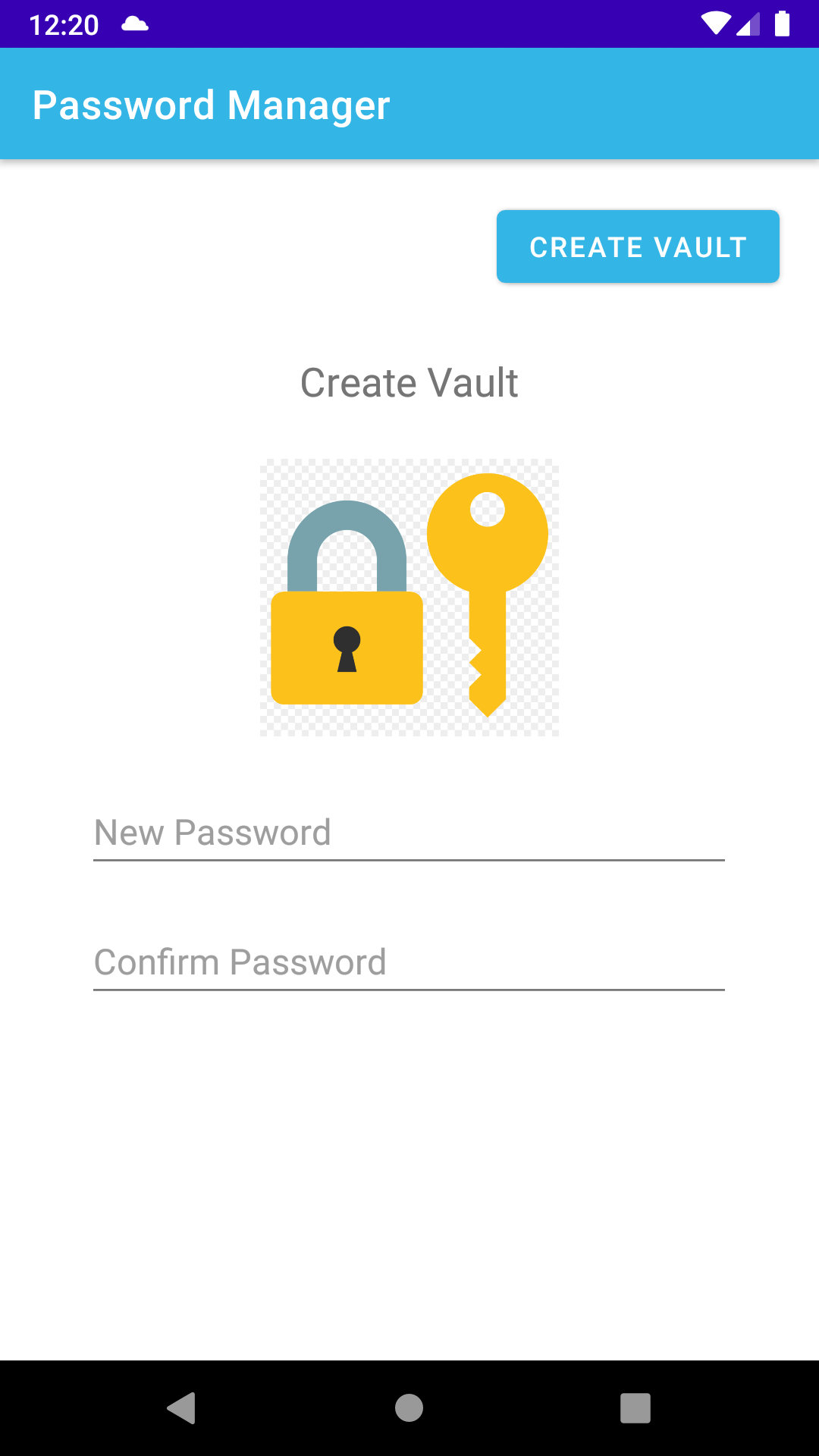
package.json

{  
 "name": "backend",  
 "version": "1.0.0",  
 "description": "",  
 "main": "index.js",  
 "scripts": {  
 "test": "echo \"Error: no test specified\" && exit 1"  
 },  
 "author": "Ebenezer Isaac",  
 "license": "ISC",  
 "dependencies": {  
 "@google-cloud/kms": "^2.8.1",  
 "dotenv": "^10.0.0",  
 "express": "^4.17.1"  
 },  
 "devDependencies": {  
 "constants": "0.0.2",  
 "cors": "^2.8.5",  
 "fast-crc32c": "^2.0.0"  
 }  
}

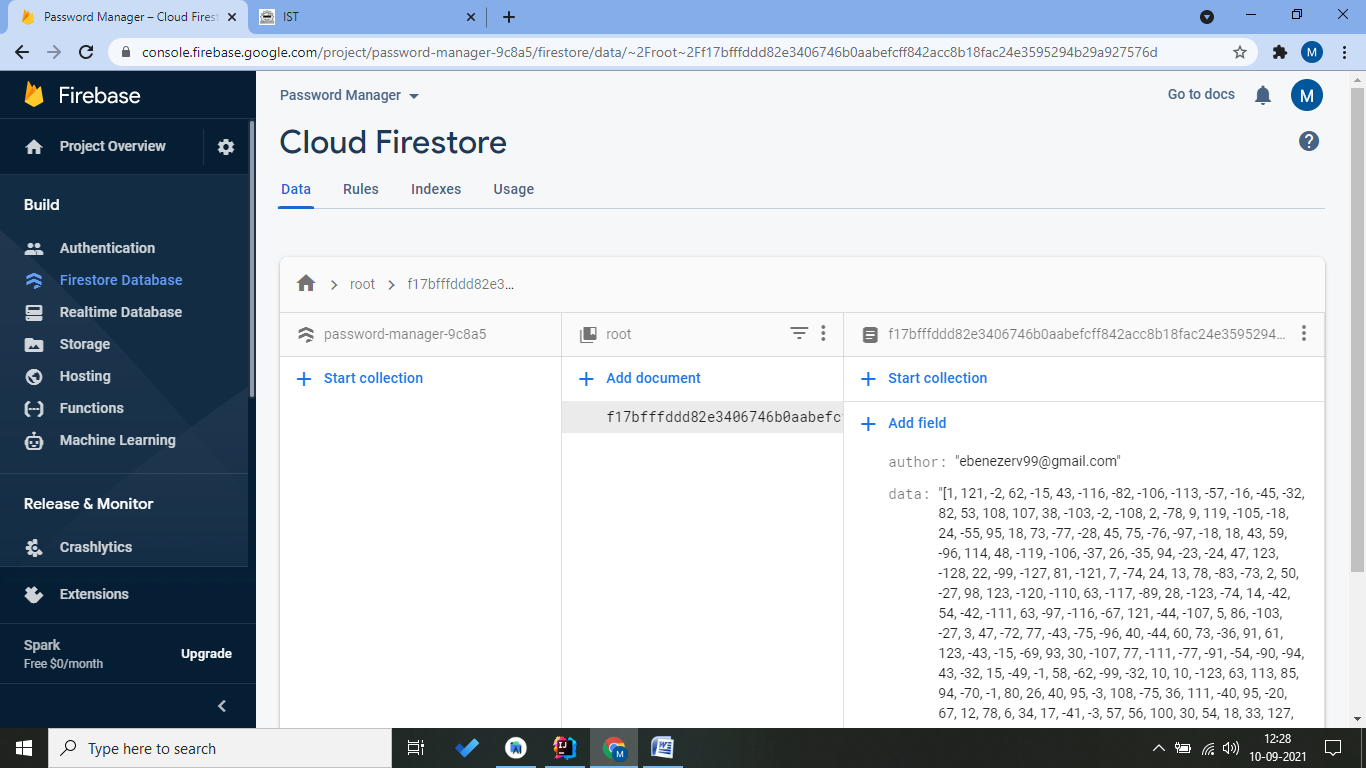
**Screenshots:**

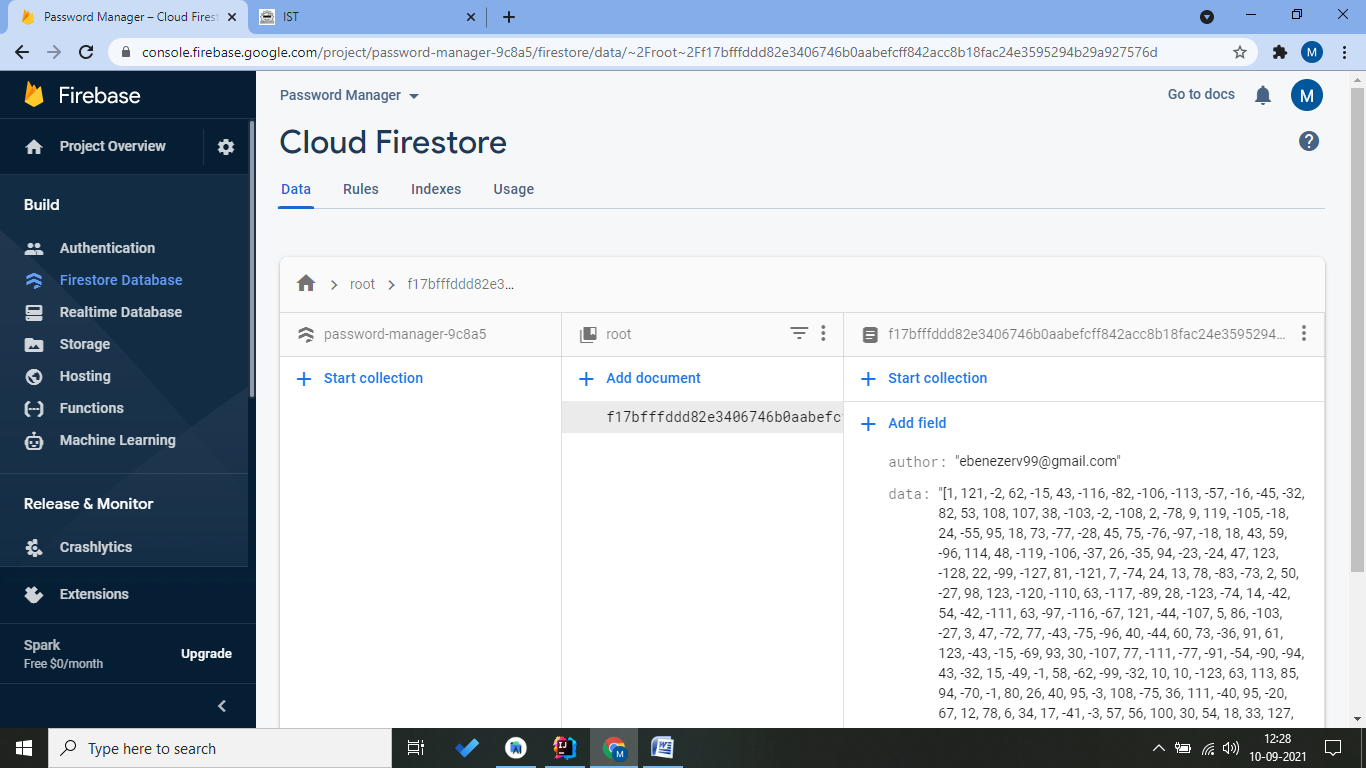


****



**Firebase Console:**

****

****