**UNIVERSITI TUNKU ABDUL RAHMAN**

**Faculty of Information and Communication Technology**



**UCCD3223 Mobile Applications Development**

**(January 2022 Trimester)**

**Individual Practical Assignment**

|  |  |
| --- | --- |
| **Name** | Tan Xi En |
| **Student ID** | 19ACB04098 |
| **Course** | Computer Science |
| **Practical Group** | P12 |
| **Lecturer** | Tan Chiang Kang |

|  |  |  |
| --- | --- | --- |
| **Marking scheme** | **Marks** | **Remarks** |
| Correctness of output | **× 3** |  |
| Design of UI | **× 2.5** |  |
| User Friendliness | **× 2.5** |  |
| Neat Program Documentation |  |  |
| Report Format |  |  |
| **TOTAL** |  |  |

**Introduction**

With the advancement of modern technology, digital transformation is altering every aspect of how today’s business operate and complete. Internet services are being introduced at every second, and these internet services requires a certain level of security to prevent unauthorized access to these services. Password is used as a mean to protect user security and prevent unauthorized access. However, user-created password may either be too simple or too repetitive, as users may use the same password for each internet service. To combat this problem, password managers are introduced to manage password for different application. Therefore, for this assignment, I plan to propose a password manager that stores, generate and manage multiple passwords for multiple applications.

**Problem Statement**

1. **Password do not have strong security**

The main issues with a lot of passwords are that they are too simple. User generated password usually comprises of only number and alphabets. These passwords are too simple to be cracked as they have a low number of combinations. For a security tool used to prevent unauthorized access to important services, passwords need to meet a certain degree of security. Therefore, our password manager will generate strong combinations of password with certain length.

1. **Some Passwords are repetitive**

The main issue with user defined passwords are that user tend to use only one password for each internet service. This is because it is much easier memorising one passwords, rather multiple unique passwords for each internet service. A user should ideally have unique passwords for all of one’s services to reduce the impact of compromised passwords. The implied effect is that if one of the services is compromised, none of the user’s other services are.

**App Functionality**

While there are many instances of password managers in Google play Store (e.g. KeePass, DashLane etc.), there are universally agreed functionalities for a password manager. A basic password manager should be able to:

* Generate Secure password with multiple character sets and variable length
* Able to add, update, delete password for multiple websites
* Encrypt password when stored into database, decrypt password when retrieved from database.
* Search and sort list of passwords.
* Export Database as Backup

To store and retrieve password, we need to create a local database within our android phone. Normally, for application development, developers would use SQLite database, as it is provided in all android phone. However, for ease of development, I’ve decided to use Room Database to implement our database functionality. This is because the code are much cleaner and it is much easier to debug using Room Database.

Secondly, to show our list of password, I have also decided to implement RecycleView adapter. This is because RecycleView have its own view model and life cycle that constantly recycle old view and reuse for new elements, hence the name recycle view. Recycle view allows our android app to run smoother, by constantly recycle old elements and remove unnecessary view.

To implement these functionalities, I have designed multiple activities and fragment that accommodates these functions. The UIs that I have designed are, activity\_main.xml, fragment\_gen\_pwd.xml, fragment\_change\_mas\_pwd.xml, activity\_login.xml, activity\_add\_pwd.xml, activity\_edit\_pwd.xml.

Login Page

|  |  |
| --- | --- |
|  |  |
| Default Login Page for Password Manager  Have eye icon button to show or hide master password. | Shows toast message if master password is incorrect. |

Main Page

|  |  |
| --- | --- |
|  |  |
| Default Homepage for Password Manager. Shows list of password that users can edit.  Shows Floating Action Button that users can press to add new password  Use Recyle View Adapter to show List of password. | Custom Navigation Drawer on the right. When user press on the navigation icon, the drawer will pop up and allow user to navigate to different sections of the application |

Generate Password Page and Change Master Password Page

|  |  |
| --- | --- |
| Generate Password Page | Change Master Password Page |
|  |  |
| Helps user to auto generate password based on selected character Sets.  Offers 4 different types of Character set, that is Uppercase Characters, Lowercase Characters, Digit Characters, Symbol Characters  Allow User to generate password on variable length  Have copy to clipboard button that user may press to copy to clipboard  Have reset button that user may press to generate a new password | Allow User to change Master Password  Shows incorrect toast message if password and confirm password are not the same  Shows success toast message if password and confirm password are the same.  Have eye icon to hide or show password |

Add Password Page

|  |  |
| --- | --- |
|  |  |
| The following activity comprises of the basic property of each password. Title, Username, Password and Website.  If title, username, passwords and website are not filled in, an incorrect toast message will be shown to act as validation.  Optional field such as Emergency PIN, security questions are provided. User may choose to fill in.  Have eye icon to hide or show password field  Have save Icon to save password | |

Edit Password Page

|  |  |  |
| --- | --- | --- |
|  |  |  |
| The following edit page provides the list of field based on the password element selected.  There are three variations of Edit Page. This is based on the number of optional fields that the user have entered. For example, “Facebook” does not contain any optional field, whereas “Snapchat” and “Amazon” does.  Each of the field have a copy clipboard button, that copies the respective field into the clipboard  Each password field have a show / hide eye icon, that shows or hide password.  Each edit page have a delete and save button function, that performs its respective function. | | |

**Encrypted Password**

One of the main priorities of a password function is its ability to encrypt password. Encrypted password is important because in the scenario the original database is hacked, hackers would be unable to get the password, as they are not stored in plaintext. To achieve this, we have decided to use AES encryption with a randomly generated key that is stored within the user shared preference file. To demonstrate the functionality, we have provided two screenshot.

Graphical user interface, application

Description automatically generated

Using app inspection (tool that is provided in android studio), we are able to debug the information in our room database. As we can observe, there is a random cipher that is generate from our AES encryption. Therefore, we are successful in storing encrypted password in our Room Database model.

Text

Description automatically generated

**AES Key:**

The randomly generated key is stored in our shared preference file. The key will be retrieved and used when the password manager is required to decrypt the password.

Text

Description automatically generated

**Exporting Database**

By selecting our option menu, users are provided with the following option to export database. This option will create “PwdDb.txt”, that stores all the password in our database. This text file can be used as a backup, in the event the application is uninstalled for unspecified reason. User can then refer to the backup text file for their list of passwords stored.

|  |  |
| --- | --- |
|  |  |

The contents of “PwdDb.txt” is shown below:

Text

Description automatically generated with medium confidence

**Conclusion**

In conclusion, our password manager application comprises of all the basic functionalities needed. This password manager app is able to accommodate for variety of websites and generate strong password up to the length of 50. It provides 4 characters sets, which is sufficient security to not be cracked easily. Furthermore, each password are encrypted using AES database and a randomly generated key unique for each android phone. Therefore, our password manager app meets the basic requirements for a functioning password manager.

**Appendix:**

**PwdListAdapter.java**

public class PwdListAdapter extends RecyclerView.Adapter<PwdListAdapter.PwdViewHolder> implements Filterable {

    public static final String TAG = PwdListAdapter.class.getSimpleName();

    private List<PasswordElem> mPwdList;

    private List<PasswordElem> mPwdListAll;

    private LayoutInflater mInflater;

    private String[] colors;

    public PwdListAdapter(Context mContext) {

        this.mPwdList = new ArrayList<PasswordElem>();

        this.mPwdListAll = new ArrayList<PasswordElem>();

        this.mInflater = LayoutInflater.from(mContext);

        this.colors = mContext.getResources().getStringArray(R.array.icon\_colors);

    }

    @NonNull

    @Override

    public PwdListAdapter.PwdViewHolder onCreateViewHolder(ViewGroup parent, int position) {

        View mItemView = mInflater.inflate(R.layout.pwd\_list\_item, parent, false);

        return new PwdViewHolder(mItemView, this);

    }

    @Override

    public void onBindViewHolder(PwdListAdapter.PwdViewHolder viewHolder, int position) {

        PasswordElem curPwd = mPwdList.get(position);

        // Update Icon TextView

        int f\_val = (int) curPwd.title.toUpperCase().charAt(0);

        int ind = (int) ((f\_val - 65) / 26.0 \* colors.length);

        viewHolder.icon\_item\_txt.getBackground().setColorFilter(Color.parseColor(colors[ind]), PorterDuff.Mode.SRC\_ATOP);

        viewHolder.icon\_item\_txt.setText(curPwd.title.substring(0, 1).toUpperCase());

        // Bind Password Element to RecycleView List Item

        viewHolder.title\_item\_txt.setText(curPwd.title);

        viewHolder.name\_item\_txt.setText(curPwd.username);

        viewHolder.website\_item\_txt.setText(curPwd.website);

    }

    @Override

    public int getItemCount() {

        return mPwdList.size();

    }

    public void updateList(List<PasswordElem> pwdList) {

        PwdListDiffCallback diffCallback = new PwdListDiffCallback(this.mPwdList, pwdList);

        DiffUtil.DiffResult diffResult = DiffUtil.calculateDiff(diffCallback);

        this.mPwdList.clear();

        this.mPwdList.addAll(pwdList);

        this.mPwdListAll.clear();

        this.mPwdListAll.addAll(pwdList);

        diffResult.dispatchUpdatesTo(this);

    }

    @Override

    public Filter getFilter() {

        return Searched\_Filter;

    }

    private Filter Searched\_Filter = new Filter() {

        @Override

        protected FilterResults performFiltering(CharSequence charSequence) {

            String charString = charSequence.toString();

            List<PasswordElem> filteredList = new ArrayList<>();

            if (charString.isEmpty()) {

                filteredList.addAll(mPwdListAll);

            } else {

                for (PasswordElem pwd : mPwdListAll) {

                    if (pwd.getTitle().toLowerCase().contains(charString.toLowerCase())) {

                        filteredList.add(pwd);

                    }

                }

            }

            FilterResults filterResults = new FilterResults();

            filterResults.values = filteredList;

            return filterResults;

        }

        @Override

        protected void publishResults(CharSequence constraint, FilterResults results) {

            mPwdList.clear();

            mPwdList.addAll((ArrayList) results.values);

            notifyDataSetChanged();

        }

    };

    class PwdViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener,View.OnCreateContextMenuListener{

        TextView icon\_item\_txt;

        TextView title\_item\_txt;

        TextView name\_item\_txt;

        TextView website\_item\_txt;

        PwdListAdapter mAdapter;

        public PwdViewHolder(View itemView, PwdListAdapter mAdapter) {

            super(itemView);

            icon\_item\_txt = itemView.findViewById(R.id.icon\_item\_txt);

            title\_item\_txt = itemView.findViewById(R.id.title\_item\_txt);

            name\_item\_txt = itemView.findViewById(R.id.name\_item\_txt);

            website\_item\_txt = itemView.findViewById(R.id.website\_item\_txt);

            this.mAdapter = mAdapter;

            itemView.setOnClickListener(this);

            itemView.setOnCreateContextMenuListener(this);

        }

        @Override

        public void onClick(View view) {

            // Get the position of the item that was clicked.

            int mPosition = getLayoutPosition();

            PasswordElem pwd = mPwdList.get(mPosition);

            // Spawn Edit page Activity

            Intent intent = new Intent(view.getContext(), EditPwdActivity.class);

            intent.putExtra("id", pwd.getId());

            view.getContext().startActivity(intent);

        }

        public void onCreateContextMenu(ContextMenu contextMenu, View view, ContextMenu.ContextMenuInfo contextMenuInfo) {

            int pos = getAdapterPosition();

            PasswordElem pwd = mPwdList.get(pos);

            MenuItem delete = contextMenu.add(Menu.NONE, 1, (int) pwd.getId(), "Delete");

        }

    }

}

**PwdListDiffCallback.java**

public class PwdListDiffCallback extends DiffUtil.Callback{

    private List<PasswordElem> oldList;

    private List<PasswordElem> newList;

    public PwdListDiffCallback(List<PasswordElem> oldList, List<PasswordElem> newList) {

        this.oldList = oldList;

        this.newList = newList;

    }

    @Override

    public int getOldListSize() {

        return this.oldList.size();

    }

    @Override

    public int getNewListSize() {

        return this.newList.size();

    }

    @Override

    public boolean areItemsTheSame(int oldItemPosition, int newItemPosition) {

        PasswordElem oldPwd = oldList.get(oldItemPosition);

        PasswordElem newPwd = newList.get(newItemPosition);

        return oldPwd.id == newPwd.id;

    }

    @Override

    public boolean areContentsTheSame(int oldItemPosition, int newItemPosition) {

        PasswordElem oldPwd = oldList.get(oldItemPosition);

        PasswordElem newPwd = newList.get(newItemPosition);

        return oldPwd.equals(newPwd);

    }

}

**PwdListViewModel.java**

public class PwdListViewModel extends AndroidViewModel {

    private PasswordElemRepository mRepository;

    private LiveData<List<PasswordElem>> pwdList;

    public PwdListViewModel(@NonNull Application app) {

        super(app);

        mRepository = new PasswordElemRepository(app);

    }

    public LiveData<List<PasswordElem>> getPwdList(){

        if(pwdList == null){pwdList = mRepository.getPasswordElemList();}

        return pwdList;

    }

    public LiveData<List<PasswordElem>> getPwdListDesc(){

        return mRepository.getPasswordElemListDesc();

    }

    public LiveData<List<PasswordElem>> getPwdListDate(){

        return mRepository.getPasswordElemListDate();

    }

    public LiveData<List<PasswordElem>> getPwdListDateDesc(){

        return mRepository.getPasswordElemListDateDesc();

    }

    public PasswordElem getPassword(long id) throws ExecutionException, InterruptedException {return mRepository.getPassword(id);}

    public void insertPassword(PasswordElem pwd) {mRepository.insertPassword(pwd);}

    public void updatePassword(PasswordElem pwd){mRepository.updatePassword(pwd);}

    public void deletePassword(PasswordElem pwd){mRepository.deletePassword(pwd);}

    public void deleteAllPassword() { mRepository.deleteAllPassword(); }

}

**AppDatabase.java**

@Database(entities = {PasswordElem.class}, version = 2, exportSchema = false)

@TypeConverters(Converters.class)

public abstract class AppDatabase extends RoomDatabase {

    public abstract PasswordElemDao pwdElemDao();

    private static AppDatabase instance = null;

    private static AppDatabase buildDatabase(Context context) {

        return Room.databaseBuilder(context.getApplicationContext(),

                AppDatabase.class, "password\_db")

                // Wipes and rebuilds instead of migrating

                // if no Migration object.

                // Migration is not part of this practical.

                .fallbackToDestructiveMigration()

                .build();

    }

    public static AppDatabase getDatabase(Context context){

        if(instance == null){

            synchronized (AppDatabase.class){

                if(instance == null)

                    instance = buildDatabase(context);

            }

        }

        return instance;

    }

}

**Converters.java**

public class Converters {

    @TypeConverter

    public long calendarToDatestamp(Calendar calendar) {

        return calendar.getTimeInMillis();

    }

    @TypeConverter

    public Calendar datestampToCalendar(long value) {

        Calendar calendar = Calendar.getInstance();

        calendar.setTimeInMillis(value);

        return calendar;

    }

}

**PasswordElem.java**

@Entity(tableName = "password\_tbl")

public class PasswordElem {

    @PrimaryKey(autoGenerate = true)

    @ColumnInfo(name = "id")

    public long id;

    @ColumnInfo(name = "title")

    public String title;

    @ColumnInfo(name = "username")

    public String username;

    @ColumnInfo(name = "password")

    public String password;

    @ColumnInfo(name = "website")

    public String website;

    @ColumnInfo(name = "pin\_number")

    public String pin\_number;

    @ColumnInfo(name = "security\_question\_1")

    public String security\_question\_1;

    @ColumnInfo(name = "security\_answer\_1")

    public String security\_answer\_1;

    @ColumnInfo(name = "security\_question\_2")

    public String security\_question\_2;

    @ColumnInfo(name = "security\_answer\_2")

    public String security\_answer\_2;

    @ColumnInfo(name = "security\_question\_3")

    public String security\_question\_3;

    @ColumnInfo(name = "security\_answer\_3")

    public String security\_answer\_3;

    @ColumnInfo(name = "created\_date")

    public Calendar created\_date;

    @ColumnInfo(name = "last\_updated\_date")

    public Calendar last\_updated\_date;

    public PasswordElem(){

        this.title = "";

        this.username = "";

        this.password = "";

        this.website = "";

        this.pin\_number = "";

        this.security\_question\_1 = "";

        this.security\_question\_2 = "";

        this.security\_answer\_3 = "";

        this.security\_answer\_1 = "";

        this.security\_answer\_2 = "";

        this.security\_answer\_3 = "";

        this.created\_date = Calendar.getInstance();

        this.last\_updated\_date = Calendar.getInstance();

    }

    public long getId() {

        return id;

    }

    public void setId(long id) {

        this.id = id;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String username) {

        this.username = username;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

    public String getWebsite() {

        return website;

    }

    public void setWebsite(String website) {

        this.website = website;

    }

    public Calendar getCreated\_date() {

        return created\_date;

    }

    public void setCreated\_date(Calendar created\_date) {

        this.created\_date = created\_date;

    }

    public Calendar getLast\_updated\_date() {

        return last\_updated\_date;

    }

    public void setLast\_updated\_date(Calendar last\_updated\_date) {

        this.last\_updated\_date = last\_updated\_date;

    }

    public String getPin\_number() {

        return pin\_number;

    }

    public void setPin\_number(String pin\_number) {

        this.pin\_number = pin\_number;

    }

    public String getSecurity\_question\_1() {

        return security\_question\_1;

    }

    public void setSecurity\_question\_1(String security\_question\_1) {

        this.security\_question\_1 = security\_question\_1;

    }

    public String getSecurity\_answer\_1() {

        return security\_answer\_1;

    }

    public void setSecurity\_answer\_1(String security\_answer\_1) {

        this.security\_answer\_1 = security\_answer\_1;

    }

    public String getSecurity\_question\_2() {

        return security\_question\_2;

    }

    public void setSecurity\_question\_2(String security\_question\_2) {

        this.security\_question\_2 = security\_question\_2;

    }

    public String getSecurity\_answer\_2() {

        return security\_answer\_2;

    }

    public void setSecurity\_answer\_2(String security\_answer\_2) {

        this.security\_answer\_2 = security\_answer\_2;

    }

    public String getSecurity\_question\_3() {

        return security\_question\_3;

    }

    public void setSecurity\_question\_3(String security\_question\_3) {

        this.security\_question\_3 = security\_question\_3;

    }

    public String getSecurity\_answer\_3() {

        return security\_answer\_3;

    }

    public void setSecurity\_answer\_3(String security\_answer\_3) {

        this.security\_answer\_3 = security\_answer\_3;

    }

    public void updateValue(String[] arr){

        this.title = arr[0];

        this.username = arr[1];

        this.password = arr[2];

        this.website = arr[3];

        this.last\_updated\_date = Calendar.getInstance();

    }

    public void updateOptionalValue(String[] arr){

        this.pin\_number = arr[0];

        this.security\_question\_1 = arr[1];

        this.security\_answer\_1 = arr[2];

        this.security\_question\_2 = arr[3];

        this.security\_answer\_2 = arr[4];

        this.security\_question\_3 = arr[5];

        this.security\_answer\_3 = arr[6];

        this.last\_updated\_date = Calendar.getInstance();

    }

    @Override

    public boolean equals(Object o) {

        if (this == o) return true;

        if (o == null || getClass() != o.getClass()) return false;

        PasswordElem that = (PasswordElem) o;

        return id == that.id

                && title.equals(that.title)

                && username.equals(that.username)

                && password.equals(that.password)

                && website.equals(that.website);

    }

    @Override

    public int hashCode() {

        return Objects.hash(id, title, username, password, website, created\_date, last\_updated\_date);

    }

    @Override

    public String toString() {

        return "id=" + id +

                ", title='" + title + '\'' +

                ", username='" + username + '\'' +

                ", password='" + password + '\'' +

                ", website='" + website + '\'' +

                ", created\_date=" + created\_date +

                ", last\_updated\_date=" + last\_updated\_date;

    }

}

**PasswordElemDao.java**

@Dao

public interface PasswordElemDao {

    @Query("SELECT \* FROM password\_tbl ORDER BY title")

    LiveData<List<PasswordElem>> getPasswordElemList();

    @Query("SELECT \* FROM password\_tbl ORDER BY title DESC")

    LiveData<List<PasswordElem>> getPasswordElemListDesc();

    @Query("SELECT \* FROM password\_tbl ORDER BY created\_date DESC")

    LiveData<List<PasswordElem>> getPasswordElemListDate();

    @Query("SELECT \* FROM password\_tbl ORDER BY created\_date")

    LiveData<List<PasswordElem>> getPasswordElemListDateDesc();

    @Query("SELECT \* FROM password\_tbl WHERE id = :pwd\_id")

    PasswordElem getPasswordElem(long pwd\_id);

    @Insert

    long insertPassword(PasswordElem pwd);

    @Update

    void updatePassword (PasswordElem pwd);

    @Delete

    void deletePassword (PasswordElem pwd);

    @Query("DELETE FROM password\_tbl")

    void deleteAllPassword();

}

**PasswordElemRepository.java**

public class PasswordElemRepository {

    private PasswordElemDao mPasswordElemDao;

    private LiveData<List<PasswordElem>> passwordElemList;

    public PasswordElemRepository(Application app){

        AppDatabase db = AppDatabase.getDatabase(app);

        mPasswordElemDao = db.pwdElemDao();

        passwordElemList = mPasswordElemDao.getPasswordElemList();

    }

    public LiveData<List<PasswordElem>> getPasswordElemList() {

        return passwordElemList;

    }

    public LiveData<List<PasswordElem>> getPasswordElemListDesc() {

        return mPasswordElemDao.getPasswordElemListDesc();

    }

    public LiveData<List<PasswordElem>> getPasswordElemListDate() {

        return mPasswordElemDao.getPasswordElemListDate();

    }

    public LiveData<List<PasswordElem>> getPasswordElemListDateDesc() {

        return mPasswordElemDao.getPasswordElemListDateDesc();

    }

    public PasswordElem getPassword(long pwdId) throws ExecutionException, InterruptedException {

        return new getPasswordAsync(mPasswordElemDao).execute(pwdId).get();

    }

    public void insertPassword (PasswordElem pwd) {

        new insertPasswordAsync(mPasswordElemDao).execute(pwd);

    }

    public void updatePassword(PasswordElem pwd){

        new updatePasswordAsync(mPasswordElemDao).execute(pwd);

    }

    public void deletePassword(PasswordElem pwd){

        new deletePasswordAsync(mPasswordElemDao).execute(pwd);

    }

    public void deleteAllPassword(){

        new deleteAllPasswordAsync(mPasswordElemDao).execute();

    }

    private static class getPasswordAsync extends AsyncTask<Long, Void, PasswordElem> {

        private PasswordElemDao mPasswordElemDaoAsync;

        public getPasswordAsync(PasswordElemDao mPasswordElemDaoAsync) {

            this.mPasswordElemDaoAsync = mPasswordElemDaoAsync;

        }

        @Override

        protected PasswordElem doInBackground(Long... ids) {

            return mPasswordElemDaoAsync.getPasswordElem(ids[0]);

        }

    }

    private static class insertPasswordAsync extends AsyncTask<PasswordElem, Void, Long>{

        private PasswordElemDao mPasswordElemDaoAsync;

        public insertPasswordAsync(PasswordElemDao mPasswordElemDaoAsync) {

            this.mPasswordElemDaoAsync = mPasswordElemDaoAsync;

        }

        @Override

        protected Long doInBackground(PasswordElem... passwordElems) {

            long id = mPasswordElemDaoAsync.insertPassword(passwordElems[0]);

            return id;

        }

    }

    private static class updatePasswordAsync extends AsyncTask<PasswordElem, Void, Void>{

        private PasswordElemDao mPasswordElemDaoAsync;

        public updatePasswordAsync(PasswordElemDao mPasswordElemDaoAsync) {

            this.mPasswordElemDaoAsync = mPasswordElemDaoAsync;

        }

        @Override

        protected Void doInBackground(PasswordElem... passwordElems) {

            mPasswordElemDaoAsync.updatePassword(passwordElems[0]);

            return null;

        }

    }

    private static class deletePasswordAsync extends AsyncTask<PasswordElem, Void, Void>{

        private PasswordElemDao mPasswordElemDaoAsync;

        public deletePasswordAsync(PasswordElemDao mPasswordElemDaoAsync) {

            this.mPasswordElemDaoAsync = mPasswordElemDaoAsync;

        }

        @Override

        protected Void doInBackground(PasswordElem... passwordElems) {

            mPasswordElemDaoAsync.deletePassword(passwordElems[0]);

            return null;

        }

    }

    private static class deleteAllPasswordAsync extends AsyncTask<Void, Void, Void>{

        private PasswordElemDao mPasswordElemDaoAsync;

        public deleteAllPasswordAsync(PasswordElemDao mPasswordElemDaoAsync) {

            this.mPasswordElemDaoAsync = mPasswordElemDaoAsync;

        }

        @Override

        protected Void doInBackground(Void... voids) {

            mPasswordElemDaoAsync.deleteAllPassword();

            return null;

        }}}

**AddPwdActivity.java**

public class AddPwdActivity extends AppCompatActivity {

    public static final String TAG = AddPwdActivity.class.getSimpleName();

    private ActivityAddPwdBinding binding;

    private PwdListViewModel viewModel;

    private EditText title\_txt;

    private EditText name\_txt;

    private EditText pwd\_txt;

    private EditText website\_txt;

    // Optional

    private EditText pinNo\_txt;

    private EditText secQes1\_txt;

    private EditText secQes2\_txt;

    private EditText secQes3\_txt;

    private EditText secAns1\_txt;

    private EditText secAns2\_txt;

    private EditText secAns3\_txt;

    private ImageView show\_pass\_btn;

    private SharedPreferences pref;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        binding = ActivityAddPwdBinding.inflate(getLayoutInflater());

        setContentView(binding.getRoot());

        // Binding Java Element to XML

        title\_txt = binding.titleTxt;

        name\_txt = binding.nameTxt;

        pwd\_txt = binding.pwdTxt;

        website\_txt = binding.websiteTxt;

        // Optional Elements

        pinNo\_txt = binding.pinNoTxt;

        secQes1\_txt = binding.secQes1Txt;

        secQes2\_txt = binding.secQes2Txt;

        secQes3\_txt = binding.secQes3Txt;

        secAns1\_txt = binding.secAns1Txt;

        secAns2\_txt = binding.secAns2Txt;

        secAns3\_txt = binding.secAns3Txt;

        show\_pass\_btn = binding.showPassBtn;

        pref = this.getSharedPreferences("mySharedPreferences", MODE\_PRIVATE);

        // Initialize Objects

        viewModel = ViewModelProviders.of(AddPwdActivity.this).get(PwdListViewModel.class);

        // Add Back Button at ActionBar

        if (getSupportActionBar() != null) {

            getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        }

        show\_pass\_btn.setOnClickListener(v -> ShowHidePass(v));

    }

    @Override

    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.menu\_add, menu);

        return true;

    }

    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();

        if (id == R.id.option\_save) {

            String pwd\_str = pwd\_txt.getText().toString();

            String key = pref.getString("decrypt\_key", "");

            // Encrypt Password using AES Encryption

            try {

                pwd\_str = util.encrypt(pwd\_str, key);

            } catch (Exception e) {

                e.printStackTrace();

            }

            String[] txt\_arr = {

                    title\_txt.getText().toString(),

                    name\_txt.getText().toString(),

                    pwd\_str,

                    website\_txt.getText().toString()

            };

            // Validation => Check to ensure Edit Text Entry are not empty

            // If There is Empty Text Field

            if(util.checkIfAnyEmpty(txt\_arr)) {

                Toast.makeText(this, "Please ensure that all textFields are not empty!", Toast.LENGTH\_SHORT).show();

                return super.onOptionsItemSelected(item);

            }

            String[] txt\_optional\_arr = {

                    pinNo\_txt.getText().toString().isEmpty() ? "" : pinNo\_txt.getText().toString(),

                    secQes1\_txt.getText().toString().isEmpty() ? "" : secQes1\_txt.getText().toString(),

                    secAns1\_txt.getText().toString().isEmpty() ? "" : secAns1\_txt.getText().toString(),

                    secQes2\_txt.getText().toString().isEmpty() ? "" : secQes2\_txt.getText().toString(),

                    secAns2\_txt.getText().toString().isEmpty() ? "" : secAns2\_txt.getText().toString(),

                    secQes3\_txt.getText().toString().isEmpty() ? "" : secQes3\_txt.getText().toString(),

                    secAns3\_txt.getText().toString().isEmpty() ? "" : secAns3\_txt.getText().toString()

            };

            PasswordElem pwd = new PasswordElem();

            pwd.updateValue(txt\_arr);

            pwd.updateOptionalValue(txt\_optional\_arr);

            viewModel.insertPassword(pwd);

            Toast.makeText(this, "Successfully Save Record!", Toast.LENGTH\_SHORT).show();

            finish();

        }

        return super.onOptionsItemSelected(item);

    }

    public void ShowHidePass(View view){

        if(pwd\_txt.getTransformationMethod().equals(PasswordTransformationMethod.getInstance())){

            show\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_24);

            //Show Password

            pwd\_txt.setTransformationMethod(HideReturnsTransformationMethod.getInstance());

        }

        else{

            show\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_off\_24);

            //Hide Password

            pwd\_txt.setTransformationMethod(PasswordTransformationMethod.getInstance());

        }

    }

    @Override

    public boolean onSupportNavigateUp() {

        finish();

        return true;

    }

}

**ChangeMasPwdFragment.java**

public class ChangeMasPwdFragment extends Fragment {

    private FragmentChangeMasPwdBinding binding;

    private EditText pwd\_txt;

    private EditText con\_pwd\_txt;

    private ImageView show\_pass\_btn;

    private ImageView show\_con\_pass\_btn;

    private Button submit\_btn;

    @Nullable

    @Override

    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

        binding = FragmentChangeMasPwdBinding.inflate(inflater, container, false);

        View root = binding.getRoot();

        // Bind Jave Elements to XML

        pwd\_txt = binding.pwdTxt;

        con\_pwd\_txt = binding.conPwdTxt;

        show\_pass\_btn = binding.showPassBtn;

        show\_con\_pass\_btn = binding.showConPassBtn;

        submit\_btn = binding.submitBtn;

        show\_pass\_btn.setOnClickListener(v -> ShowHidePass(v));

        show\_con\_pass\_btn.setOnClickListener(v -> ShowConHidePass(v));

        submit\_btn.setOnClickListener(v -> submitPassword(v));

        return root;

    }

    public void submitPassword(View v){

        String pwd = pwd\_txt.getText().toString();

        String con\_pwd = con\_pwd\_txt.getText().toString();

        if(pwd.equals(con\_pwd)){

            pwd\_txt.setText("");

            con\_pwd\_txt.setText("");

            SharedPreferences pref = v.getContext().getSharedPreferences("mySharedPreferences", MODE\_PRIVATE);

            SharedPreferences.Editor prefEditor = pref.edit();

            prefEditor.putString("master\_password", pwd);

            prefEditor.commit();

            Toast.makeText(v.getContext(), String.format("Master Password Successfully Updated!"), Toast.LENGTH\_SHORT).show();

        } else{

            pwd\_txt.setText("");

            con\_pwd\_txt.setText("");

            Toast.makeText(v.getContext(), String.format("Both passwords are not the same!\nPlease try again."), Toast.LENGTH\_SHORT).show();

        }

    }

    public void ShowHidePass(View view){

        if(pwd\_txt.getTransformationMethod().equals(PasswordTransformationMethod.getInstance())){

            show\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_24);

            //Show Password

            pwd\_txt.setTransformationMethod(HideReturnsTransformationMethod.getInstance());

        }

        else{

            show\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_off\_24);

            //Hide Password

            pwd\_txt.setTransformationMethod(PasswordTransformationMethod.getInstance());

        }

    }

    public void ShowConHidePass(View view){

        if(con\_pwd\_txt.getTransformationMethod().equals(PasswordTransformationMethod.getInstance())){

            show\_con\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_24);

            //Show Password

            con\_pwd\_txt.setTransformationMethod(HideReturnsTransformationMethod.getInstance());

        }

        else{

            show\_con\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_off\_24);

            //Hide Password

            con\_pwd\_txt.setTransformationMethod(PasswordTransformationMethod.getInstance());

        }

    }

    @Override

    public void onDestroyView() {

        super.onDestroyView();

        binding = null;

    }

}

**EditPwdActivity.java**

public class EditPwdActivity extends AppCompatActivity {

    public static final String TAG = EditPwdActivity.class.getSimpleName();

    private ActivityEditPwdBinding binding;

    private PwdListViewModel viewModel;

    private PasswordElem curPwd;

    private EditText title\_txt;

    private EditText name\_txt;

    private EditText pwd\_txt;

    private EditText website\_txt;

    private ImageView show\_pass\_btn;

    private ImageView title\_copy\_btn;

    private ImageView name\_copy\_btn;

    private ImageView pwd\_copy\_btn;

    private ImageView website\_copy\_btn;

    // Optional Elements

    private EditText pinNo\_txt;

    private TextView secQes1\_txtView;

    private TextView secQes2\_txtView;

    private TextView secQes3\_txtView;

    private EditText secAns1\_txt;

    private EditText secAns2\_txt;

    private EditText secAns3\_txt;

    private ImageView pinNo\_copy\_btn;

    private ImageView secAns1\_copy\_btn;

    private ImageView secAns2\_copy\_btn;

    private ImageView secAns3\_copy\_btn;

    private LinearLayout pin\_ll;

    private LinearLayout sec\_1\_ll;

    private LinearLayout sec\_2\_ll;

    private LinearLayout sec\_3\_ll;

    private SharedPreferences pref;

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        binding = ActivityEditPwdBinding.inflate(getLayoutInflater());

        setContentView(binding.getRoot());

        // Add Back Button at ActionBar

        if (getSupportActionBar() != null) {

            getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        }

        // Initialize Objects

        viewModel = ViewModelProviders.of(EditPwdActivity.this).get(PwdListViewModel.class);

        // Binding Java Element to XML

        title\_txt = binding.titleTxt;

        name\_txt = binding.nameTxt;

        pwd\_txt = binding.pwdTxt;

        website\_txt = binding.websiteTxt;

        show\_pass\_btn = binding.showPassBtn;

        title\_copy\_btn = binding.titleCopyBtn;

        name\_copy\_btn = binding.nameCopyBtn;

        pwd\_copy\_btn = binding.pwdCopyBtn;

        website\_copy\_btn = binding.websiteCopyBtn;

        // Optional

        pinNo\_txt = binding.pinNoTxt;

        secQes1\_txtView = binding.secQes1TxtView;

        secQes2\_txtView = binding.secQes2TxtView;

        secQes3\_txtView = binding.secQes3TxtView;

        secAns1\_txt = binding.secAns1Txt;

        secAns2\_txt = binding.secAns2Txt;

        secAns3\_txt = binding.secAns3Txt;

        pinNo\_copy\_btn = binding.pinNoCopyBtn;

        secAns1\_copy\_btn = binding.secAns1CopyBtn;

        secAns2\_copy\_btn = binding.secAns2CopyBtn;

        secAns3\_copy\_btn = binding.secAns3CopyBtn;

        pin\_ll = binding.pinLl;

        sec\_1\_ll = binding.sec1Ll;

        sec\_2\_ll = binding.sec2Ll;

        sec\_3\_ll = binding.sec3Ll;

        pref = this.getSharedPreferences("mySharedPreferences", MODE\_PRIVATE);

        Intent mIntent = getIntent();

        Long ind = (long) mIntent.getLongExtra("id", 0);

        try {

            curPwd = viewModel.getPassword(ind);

        } catch (ExecutionException e) {

            e.printStackTrace();

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

        title\_txt.setText(curPwd.getTitle());

        name\_txt.setText(curPwd.getUsername());

        // Decrypt Password

        String pwd\_str = curPwd.getPassword();

        String key = pref.getString("decrypt\_key", "");

        // Encrypt Password using AES Encryption

        try {

            pwd\_str = util.decrypt(pwd\_str, key);

        } catch (Exception e) {

            e.printStackTrace();

        }

        pwd\_txt.setText(pwd\_str);

        website\_txt.setText(curPwd.getWebsite());

        pinNo\_txt.setText(curPwd.getPin\_number());

        if(curPwd.getPin\_number().isEmpty()){

            pin\_ll.setVisibility(View.GONE);

        }

        secQes1\_txtView.setText(curPwd.getSecurity\_question\_1());

        secAns1\_txt.setText(curPwd.getSecurity\_answer\_1());

        if(curPwd.getSecurity\_question\_1().isEmpty()){

            sec\_1\_ll.setVisibility(View.GONE);

        }

        secQes2\_txtView.setText(curPwd.getSecurity\_question\_2());

        secAns2\_txt.setText(curPwd.getSecurity\_answer\_2());

        if(curPwd.getSecurity\_question\_2().isEmpty()){

            sec\_2\_ll.setVisibility(View.GONE);

        }

        secQes3\_txtView.setText(curPwd.getSecurity\_question\_3());

        secAns3\_txt.setText(curPwd.getSecurity\_answer\_3());

        if(curPwd.getSecurity\_question\_3().isEmpty()){

            sec\_3\_ll.setVisibility(View.GONE);

        }

        show\_pass\_btn.setOnClickListener(v -> ShowHidePass(v));

        // Copy to Clipboard

        title\_copy\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(

                v.getContext(), "Title has been copied to clipboard.", title\_txt.getText().toString()));

        name\_copy\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(

                v.getContext(), "Username has been copied to clipboard.", name\_txt.getText().toString()));

        pwd\_copy\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(

                v.getContext(), "Password has been copied to clipboard.", pwd\_txt.getText().toString()));

        website\_copy\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(

                v.getContext(), "URL has been copied to clipboard.", website\_txt.getText().toString()));

        pinNo\_copy\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(

                v.getContext(), "PIN Number has been copied to clipboard.", pinNo\_txt.getText().toString()));

        secAns1\_copy\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(

                v.getContext(), "Security Answer has been copied to clipboard.", secAns1\_txt.getText().toString()));

        secAns2\_copy\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(

                v.getContext(), "Security Answer has been copied to clipboard.", secAns2\_txt.getText().toString()));

        secAns3\_copy\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(

                v.getContext(), "Security Answer has been copied to clipboard.", secAns3\_txt.getText().toString()));

    }

    @Override

    public boolean onSupportNavigateUp() {

        finish();

        return true;

    }

    @Override

    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.menu\_edit, menu);

        return true;

    }

    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();

        if (id == R.id.option\_save) {

            String pwd\_str = pwd\_txt.getText().toString();

            String key = pref.getString("decrypt\_key", "");

            // Encrypt Password using AES Encryption

            try {

                pwd\_str = util.encrypt(pwd\_str, key);

            } catch (Exception e) {

                e.printStackTrace();

            }

            String[] txt\_arr = {

                    title\_txt.getText().toString(),

                    name\_txt.getText().toString(),

                    pwd\_str,

                    website\_txt.getText().toString()

            };

            // Validation => Check to ensure Edit Text Entry are not empty

            // If There is Empty Text Field

            if(util.checkIfAnyEmpty(txt\_arr)) {

                Toast.makeText(this, "Please ensure that all textFields are not empty!", Toast.LENGTH\_SHORT).show();

                return super.onOptionsItemSelected(item);

            }

            String[] txt\_optional\_arr = {

                    pinNo\_txt.getText().toString().isEmpty() ? "" : pinNo\_txt.getText().toString(),

                    secQes1\_txtView.getText().toString().isEmpty() ? "" : secQes1\_txtView.getText().toString(),

                    secAns1\_txt.getText().toString().isEmpty() ? "" : secAns1\_txt.getText().toString(),

                    secQes2\_txtView.getText().toString().isEmpty() ? "" : secQes2\_txtView.getText().toString(),

                    secAns2\_txt.getText().toString().isEmpty() ? "" : secAns2\_txt.getText().toString(),

                    secQes3\_txtView.getText().toString().isEmpty() ? "" : secQes3\_txtView.getText().toString(),

                    secAns3\_txt.getText().toString().isEmpty() ? "" : secAns3\_txt.getText().toString()

            };

            curPwd.updateValue(txt\_arr);

            curPwd.updateOptionalValue(txt\_optional\_arr);

            viewModel.updatePassword(curPwd);

            Toast.makeText(this, "Successfully Edited Record!", Toast.LENGTH\_SHORT).show();

            finish();

        } else if(id == R.id.option\_delete){

            // Alert Box

            new AlertDialog.Builder(this)

                    .setMessage("Are you sure you want to delete?")

                    .setCancelable(false)

                    .setPositiveButton("Yes", (dialog, ind) -> {

                        viewModel.deletePassword(curPwd);

                        Toast.makeText(this, "Successfully Deleted Record!", Toast.LENGTH\_SHORT).show();

                        finish();

                    })

                    .setNegativeButton("No", (dialog, ind) -> dialog.cancel())

                    .show();

        }

        return super.onOptionsItemSelected(item);

    }

    public void ShowHidePass(View view){

        if(pwd\_txt.getTransformationMethod().equals(PasswordTransformationMethod.getInstance())){

            show\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_24);

            //Show Password

            pwd\_txt.setTransformationMethod(HideReturnsTransformationMethod.getInstance());

        }

        else{

            show\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_off\_24);

            //Hide Password

            pwd\_txt.setTransformationMethod(PasswordTransformationMethod.getInstance());

        }

    }

}

**GenPwdFragment.java**

public class GenPwdFragment extends Fragment {

    private FragmentGenPwdBinding binding;

    private TextView pwd\_txtView;

    private TextView pwd\_len\_txtView;

    private ImageView reset\_pwd\_btn;

    private ImageView copy\_pwd\_btn;

    private Slider pwd\_slider;

    private SwitchMaterial uppercase\_switch;

    private SwitchMaterial lowercase\_switch;

    private SwitchMaterial digit\_switch;

    private SwitchMaterial symbol\_switch;

    private Context context;

    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

        binding = FragmentGenPwdBinding.inflate(inflater, container, false);

        View root = binding.getRoot();

        context = getContext();

        // Bind Java Object to XML Element

        pwd\_txtView = binding.pwdTxtView;

        pwd\_len\_txtView = binding.pwdLenTxtView;

        reset\_pwd\_btn = binding.resetPwdBtn;

        copy\_pwd\_btn = binding.copyPwdBtn;

        pwd\_slider = binding.pwdSlider;

        uppercase\_switch = binding.uppercaseSwitch;

        lowercase\_switch = binding.lowercaseSwitch;

        digit\_switch = binding.digitSwitch;

        symbol\_switch = binding.symbolSwitch;

        // Set Default Password

        String password = util.genPassword(4, true, true, true, true);

        pwd\_txtView.setText(password);

        // Update & Set Password Length

        pwd\_slider.addOnChangeListener((slider, value, fromUser) -> {

            int pwd\_len = (int) value;

            updatePwdAndLength(pwd\_len);

        });

        uppercase\_switch.setOnClickListener(v -> updatePwd());

        lowercase\_switch.setOnClickListener(v -> updatePwd());

        digit\_switch.setOnClickListener(v -> updatePwd());

        symbol\_switch.setOnClickListener(v -> updatePwd());

        // Check if at least one option is selected

        reset\_pwd\_btn.setOnClickListener(v -> updatePwd());

        // Copy to Clipboard

        copy\_pwd\_btn.setOnClickListener(v -> util.copyCodeInClipBoard(v.getContext(), "Password has been copied to clipboard.", pwd\_txtView.getText().toString()));

        return root;

    }

    public boolean checkOptions(boolean needUpperCase, boolean needLowerCase, boolean needDigit, boolean needSymbol){

        return needUpperCase || needLowerCase || needDigit || needSymbol;

    }

    // Get Password Length

    public void updatePwdAndLength(int pwd\_len){

        boolean needUpper = uppercase\_switch.isChecked();

        boolean needLower = lowercase\_switch.isChecked();

        boolean needDigit = digit\_switch.isChecked();

        boolean needSymbol = symbol\_switch.isChecked();

        // Check if there's at least one Option

        if(!checkOptions(needUpper, needLower, needDigit, needSymbol)){

            pwd\_txtView.setText("Not enough usable characters defined");

            return;

        }

        // Update Password Length

        pwd\_len\_txtView.setText(String.format("Password length: %d", pwd\_len));

        // Also generates New Password

        String password = util.genPassword(pwd\_len, needUpper, needLower, needDigit, needSymbol);

        pwd\_txtView.setText(password);

    }

    // Reset

    public void updatePwd(){

        boolean needUpper = uppercase\_switch.isChecked();

        boolean needLower = lowercase\_switch.isChecked();

        boolean needDigit = digit\_switch.isChecked();

        boolean needSymbol = symbol\_switch.isChecked();

        if(!checkOptions(needUpper, needLower, needDigit, needSymbol)){

            pwd\_txtView.setText("Not enough usable characters defined");

            return;

        }

        int pwd\_len = (int) pwd\_slider.getValue();

        String password = util.genPassword(pwd\_len, needUpper, needLower, needDigit, needSymbol);

        pwd\_txtView.setText(password);

    }

    @Override

    public void onDestroyView() {

        super.onDestroyView();

        binding = null;

    }

}

**LoginActivity.java**

public class LoginActivity extends AppCompatActivity {

    public static final String TAG = LoginActivity.class.getSimpleName();

    private ActivityLoginBinding binding;

    private EditText pwd\_txt;

    private Button submit\_btn;

    private ImageView show\_pass\_btn;

    private SharedPreferences pref;

    @Override

    protected void onCreate(@Nullable Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        binding = ActivityLoginBinding.inflate(getLayoutInflater());

        setContentView(binding.getRoot());

        // Bind Java Elements to XML Objects

        pwd\_txt = binding.pwdTxt;

        submit\_btn = binding.submitBtn;

        show\_pass\_btn = binding.showPassBtn;

        // Check if Saved Preference Exits

        pref = getSharedPreferences("mySharedPreferences", MODE\_PRIVATE);

        // If Master Password Does Not Exist

        if(pref.getString("master\_password", null) == null){

            SharedPreferences.Editor prefEditor = pref.edit();

            prefEditor.putString("master\_password", "root");

            prefEditor.commit();

        }

        // If Decryption Key Does not Exist

        if(pref.getString("decrypt\_key", null) == null){

            SharedPreferences.Editor prefEditor = pref.edit();

            String key = util.genPassword(16, true, true, true, false);

            prefEditor.putString("decrypt\_key", key);

            prefEditor.commit();

        }

        show\_pass\_btn.setOnClickListener(v -> ShowHidePass(v));

        // When Submit Button is clicked, the password is parsed to check if its similar or something

        submit\_btn.setOnClickListener(v -> submitPassword(v));

    }

    public void submitPassword(View v){

        String pwd = pwd\_txt.getText().toString();

        String master\_pwd = pref.getString("master\_password", null);

        if(pwd.equals(master\_pwd)){

            pwd\_txt.setText("");

            Intent intent = new Intent(this, MainActivity.class);

            startActivity(intent);

        } else{

            pwd\_txt.setText("");

            Toast.makeText(this, String.format("Password is incorrect!\nPlease try again."), Toast.LENGTH\_SHORT).show();

        }

    }

    public void ShowHidePass(View view){

        if(pwd\_txt.getTransformationMethod().equals(PasswordTransformationMethod.getInstance())){

            show\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_24);

            //Show Password

            pwd\_txt.setTransformationMethod(HideReturnsTransformationMethod.getInstance());

        }

        else{

            show\_pass\_btn.setImageResource(R.drawable.ic\_baseline\_visibility\_off\_24);

            //Hide Password

            pwd\_txt.setTransformationMethod(PasswordTransformationMethod.getInstance());

        }

    }

}

**MainActivity.java**

public class MainActivity extends AppCompatActivity {

    public static final String TAG = MainActivity.class.getSimpleName();

    private AppBarConfiguration mAppBarConfiguration;

    private ActivityMainBinding binding;

    private PwdListViewModel viewModel;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());

        setContentView(binding.getRoot());

        viewModel = ViewModelProviders.of(this).get(PwdListViewModel.class);

        setSupportActionBar(binding.appBarMain.toolbar);

        binding.appBarMain.toolbar.setOnMenuItemClickListener(item -> {

            int id = item.getItemId();

            if (id == R.id.option\_export) {

                FileOutputStream fos;

                String fileName = "PwdDb.txt";

                try {

                    fos = openFileOutput(fileName, MODE\_PRIVATE);

                    viewModel.getPwdList().observe(this, pwdList -> {

                        for (PasswordElem elem : pwdList) {

                            try {

                                fos.write(String.format("%s\n", elem.toString()).getBytes());

                            } catch (IOException e) {

                                e.printStackTrace();

                            }

                        }

                    });

                    fos.close();

                } catch (IOException e) {

                    e.printStackTrace();

                }

                Toast.makeText(this, "PwdDb.txt was successfully generated!", Toast.LENGTH\_SHORT).show();

            }

            return true;

        });

        DrawerLayout drawer = binding.drawerLayout;

        NavigationView navigationView = binding.navView;

        // Passing each menu ID as a set of Ids because each

        // menu should be considered as top level destinations.

        mAppBarConfiguration = new AppBarConfiguration.Builder(

                R.id.nav\_pwd\_list, R.id.nav\_gen\_pwd, R.id.nav\_change\_mas\_pwd)

                .setOpenableLayout(drawer)

                .build();

        NavController navController = Navigation.findNavController(this, R.id.nav\_host\_fragment\_content\_main);

        NavigationUI.setupActionBarWithNavController(this, navController, mAppBarConfiguration);

        navigationView.setNavigationItemSelectedListener(item -> {

            if (item.getItemId() == R.id.nav\_exit) {

                new AlertDialog.Builder(this)

                        .setMessage("Are you sure you want to logout?")

                        .setCancelable(false)

                        .setPositiveButton("Yes", (dialog, id) -> finish())

                        .setNegativeButton("No", (dialog, id) -> dialog.cancel())

                        .show();

            } else {

                NavigationUI.onNavDestinationSelected(item, navController);

                drawer.closeDrawers();

            }

            return false;

        });

    }

    @Override

    public boolean onSupportNavigateUp() {

        NavController navController = Navigation.findNavController(this, R.id.nav\_host\_fragment\_content\_main);

        return NavigationUI.navigateUp(navController, mAppBarConfiguration)

                || super.onSupportNavigateUp();

    }

    @Override

    public void onBackPressed() {

        new AlertDialog.Builder(this)

                .setMessage("Are you sure you want to exit?")

                .setCancelable(false)

                .setPositiveButton("Yes", (dialog, id) -> finishAffinity())

                .setNegativeButton("No", (dialog, id) -> dialog.cancel())

                .show();

    }

    @Override

    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.menu\_main, menu);

        return true;

    }

}

**PwdListFragment.java**

public class PwdListFragment extends Fragment {

    public static final String TAG = PwdListFragment.class.getSimpleName();

    private FragmentPwdListBinding binding;

    private RecyclerView mRecyclerView;

    private PwdListViewModel viewModel;

    private PwdListAdapter mAdapter;

    private SearchView searchBar;

    private ImageView sort\_btn;

    private FloatingActionButton fab;

    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

        binding = FragmentPwdListBinding.inflate(inflater, container, false);

        View root = binding.getRoot();

        mRecyclerView = binding.pwdRecView;

        searchBar = binding.searchBar;

        sort\_btn = binding.sortBtn;

        fab = binding.fab;

        fab.setOnClickListener(v -> nAddPwd(v));

        viewModel = ViewModelProviders.of(this.getActivity()).get(PwdListViewModel.class);

        mAdapter = new PwdListAdapter(this.getActivity());

        mRecyclerView.setAdapter(mAdapter);

        // Give the RecyclerView a default layout manager.

        mRecyclerView.setLayoutManager(new LinearLayoutManager(this.getActivity()));

        DividerItemDecoration dividerItemDecoration = new DividerItemDecoration(this.getActivity(), DividerItemDecoration.VERTICAL);

        mRecyclerView.addItemDecoration(dividerItemDecoration);

        // Updates Password List

        viewModel.getPwdList().observe(this, pwdList -> {

            mAdapter.updateList(pwdList);

        });

        searchBar.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

            @Override

            public boolean onQueryTextSubmit(String query) {

                return false;

            }

            @Override

            public boolean onQueryTextChange(String newText) {

                mAdapter.getFilter().filter(newText);

                return false;

            }

        });

        sort\_btn.setOnClickListener(view -> {

            PopupMenu popup = new PopupMenu(this.getActivity(), sort\_btn);

            popup.getMenuInflater().inflate(R.menu.menu\_sort, popup.getMenu());

            //registering popup with OnMenuItemClickListener

            popup.setOnMenuItemClickListener(item -> {

                int id = item.getItemId();

                if(id == R.id.option\_a\_to\_z){

                    viewModel.getPwdList().observe(this, pwdList -> mAdapter.updateList(pwdList));

                } else if(id == R.id.option\_z\_to\_a){

                    viewModel.getPwdListDesc().observe(this, pwdList -> mAdapter.updateList(pwdList));

                } else if(id == R.id.option\_new\_to\_old){

                    viewModel.getPwdListDate().observe(this, pwdList -> mAdapter.updateList(pwdList));

                } else if(id == R.id.option\_old\_to\_new){

                    viewModel.getPwdListDateDesc().observe(this, pwdList -> mAdapter.updateList(pwdList));

                }

                return true;

            });

            popup.show();//showing popup menu

        });

        return root;

    }

    @Override

    public boolean onContextItemSelected(MenuItem item) {

        int pos = item.getOrder();

        // Alert Dialog

        new AlertDialog.Builder(this.getContext())

                .setMessage("Are you sure you want to delete?")

                .setCancelable(false)

                .setPositiveButton("Yes", (dialog, id) -> {

                    PasswordElem pwd = null;

                    try {

                        pwd = viewModel.getPassword((long) pos);

                    } catch (ExecutionException e) {

                        e.printStackTrace();

                    } catch (InterruptedException e) {

                        e.printStackTrace();

                    }

                    viewModel.deletePassword(pwd);

                    viewModel.getPwdList().observe(this, pwdList -> {

                        mAdapter.updateList(pwdList);

                    });

                })

                .setNegativeButton("No", (dialog, id) -> dialog.cancel())

                .show();

        return super.onContextItemSelected(item);

    }

    @Override

    public void onResume() {

        super.onResume();

        viewModel.getPwdList().observe(this, pwdList -> {

            mAdapter.updateList(pwdList);

        });

    }

    public void nAddPwd(View v){

        Intent intent = new Intent(v.getContext(), AddPwdActivity.class);

        startActivity(intent);

    }

    @Override

    public void onDestroyView() {

        super.onDestroyView();

        binding = null;

    }

}

**Util.java**

public class util {

    private static final String ALGORITHM = "AES";

    public static boolean checkIfAnyEmpty(String[] arr){

        for(String s : arr){

            if(s.isEmpty()){

                return true;

            }

        }

        return false;

    }

    /\*\*

     \* Method to code text in clip board

     \*

     \* @param context context

     \* @param text    text what wan to copy in clipboard

     \* @param label   label what want to copied

     \*/

    public static void copyCodeInClipBoard(Context context, String label, String text) {

        if (context != null) {

            ClipboardManager clipboard = (ClipboardManager) context.getSystemService(Context.CLIPBOARD\_SERVICE);

            ClipData clip = ClipData.newPlainText(label, text);

            if (clipboard == null || clip == null)

                return;

            clipboard.setPrimaryClip(clip);

            Toast.makeText(context, "Successfully Copied to Clipboard!", Toast.LENGTH\_SHORT).show();

        }

    }

    public static String genPassword(int pwd\_len, boolean needUpperCase, boolean needLowerCase, boolean needDigit, boolean needSymbol){

        String password = genPassword2(pwd\_len, needUpperCase, needLowerCase, needDigit, needSymbol);

        while(!check(password, needUpperCase, needLowerCase, needDigit, needSymbol)){

            password = genPassword2(pwd\_len, needUpperCase, needLowerCase, needDigit, needSymbol);

        }

        return password;

    }

    public static String genPassword2(int pwd\_len, boolean needUpperCase, boolean needLowerCase, boolean needDigit, boolean needSymbol){

        String lowercase\_str = "abcdefghijklmnopqrstuvwxyz";

        String uppercase\_str = lowercase\_str.toUpperCase();

        String digit\_str = "012345689";

        String symbol\_str = "#?!:;?%\*£€$=+@{}[]&()";

        String chars="";

        if(needUpperCase) chars += uppercase\_str;

        if(needLowerCase) chars += lowercase\_str;

        if(needDigit) chars += digit\_str;

        if(needSymbol) chars += symbol\_str;

        StringBuilder finalPassword = new StringBuilder();

        for(int i = 0; i < pwd\_len; i++)  {

            int index = ThreadLocalRandom.current().nextInt(0,chars.length());

            finalPassword.append(chars.charAt(index));

        }

        return finalPassword.toString();

    }

    public static boolean isAlphaNumeric(char char1) {

        return (char1 >= 'a' && char1 <= 'z') || (char1 >= 'A' && char1 <= 'Z') || (char1 >= '0' && char1 <= '9');

    }

    private static boolean check(String password, boolean needUpperCase, boolean needLowerCase, boolean needDigit, boolean needSymbol) {

        //we declare our booleans

        boolean hasDigit = false;

        boolean hasSymbol = false;

        boolean hasLower = false;

        boolean hasUpper = false;

        for(char c : password.toCharArray()) {

            //we check that the password corresponds to a sufficient level of security according to the selected options

            if (needUpperCase && !hasUpper) hasUpper = Character.isUpperCase(c);

            if (needLowerCase && !hasLower) hasLower = Character.isLowerCase(c);

            if (needDigit && !hasDigit) hasDigit = Character.isDigit(c);

            if (needSymbol && !hasSymbol) hasSymbol = !isAlphaNumeric(c);

        }

        return (!needUpperCase || hasUpper) && (!needLowerCase || hasLower) && (!needDigit || hasDigit) && (!needSymbol || hasSymbol);

    }

    public static String encrypt(String value, String KEY) throws Exception

    {

        Key key = generateKey(KEY);

        Cipher cipher = Cipher.getInstance(ALGORITHM);

        cipher.init(Cipher.ENCRYPT\_MODE, key);

        byte [] encryptedByteValue = cipher.doFinal(value.getBytes("utf-8"));

        String encryptedValue64 = Base64.encodeToString(encryptedByteValue, Base64.DEFAULT);

        return encryptedValue64;

    }

    public static String decrypt(String value, String KEY) throws Exception

    {

        Key key = generateKey(KEY);

        Cipher cipher = Cipher.getInstance(ALGORITHM);

        cipher.init(Cipher.DECRYPT\_MODE, key);

        byte[] decryptedValue64 = Base64.decode(value, Base64.DEFAULT);

        byte [] decryptedByteValue = cipher.doFinal(decryptedValue64);

        String decryptedValue = new String(decryptedByteValue,"utf-8");

        return decryptedValue;

    }

    private static Key generateKey(String KEY) throws Exception

    {

        Key key = new SecretKeySpec(KEY.getBytes(), ALGORITHM);

        return key;

    }

}