TEAM MENTORS:

RAKESH AMRUTKAR
SAHIL PAVASKAR

V - CONNECT

A project on Android App Development
Under Community of Coders'
Inheritance 2019

INDEX

CONCEPT

A college app that has the following features:

1. Information about upcoming events of all student activities in the form of posts that can be added, edited and deleted by admin of respective committees. All users will be able to view posts.
2. Separate login for students with groups according to their batches handled by their CRs. All notices regarding assignment submissions, cancelled lectures, extended lectures, syllabus of upcoming semester, etc. in one place.
3. A 3d map of college for quick navigation.

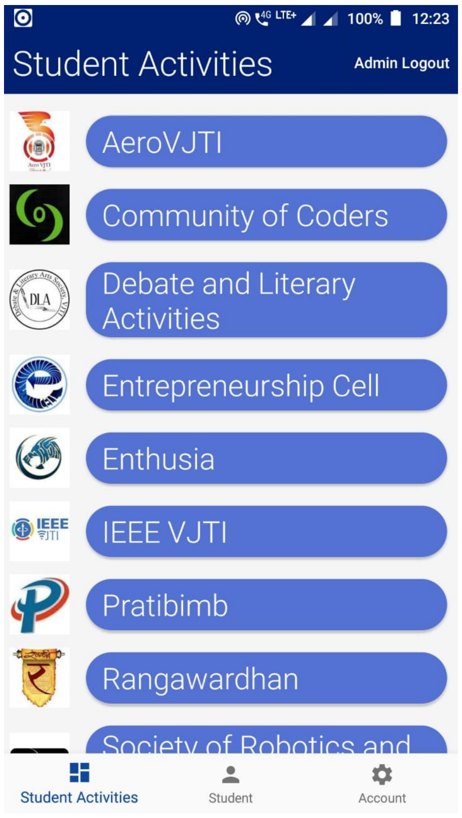
COMPLETION STATUS

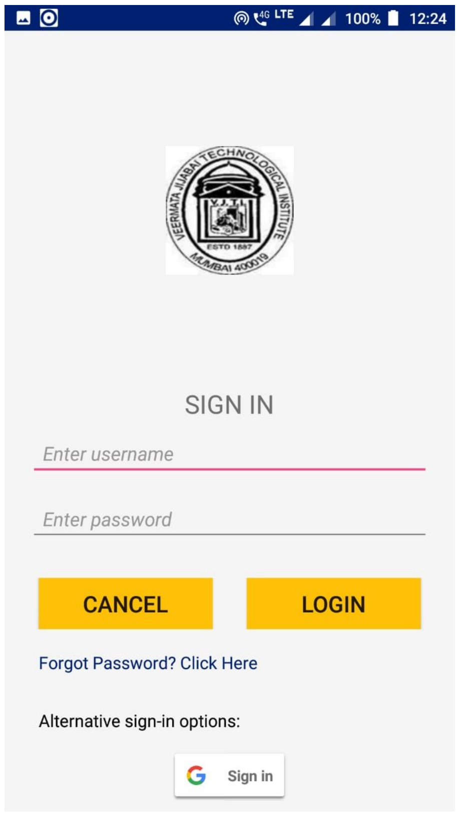
The app opens with a welcoming message and a list of all committees. At the bottom there is a navigation bar for switching between Student Activities, Student Interface and Account Settings. When the user clicks on a particular committee, he/she is directed to respective committee's feed page that consists of posts, which have a title, date, description, and an image. A login page is also given for admins of respective committees. They will be provided with an ID and a password. They can add, edit or delete posts from their respective committee's page. Students can sign up and login with their email and password. They are asked to select their program, stream and year. A separate login for CRs who can add notices in the form of posts.

SOFTWARE USED

- **Android Studio 3.4.2** an IDE for developing applications exclusively for Android platform. It has a strong editor tool for developing creative UI and emulators for different versions to test and simulate sensors without having actual Android devices.
- **Firebase** for giving users a quick, intuitive sign-in process with Firebase Authentication, and providing users with content in the form of posts, stored in Firebase Realtime Database. Another great functionality provided by Firebase is Firebase Storage, which can be used to store media and files.
- **Git**, a version control system (VCS) that allows developers to easily maintain and track changes in their projects.

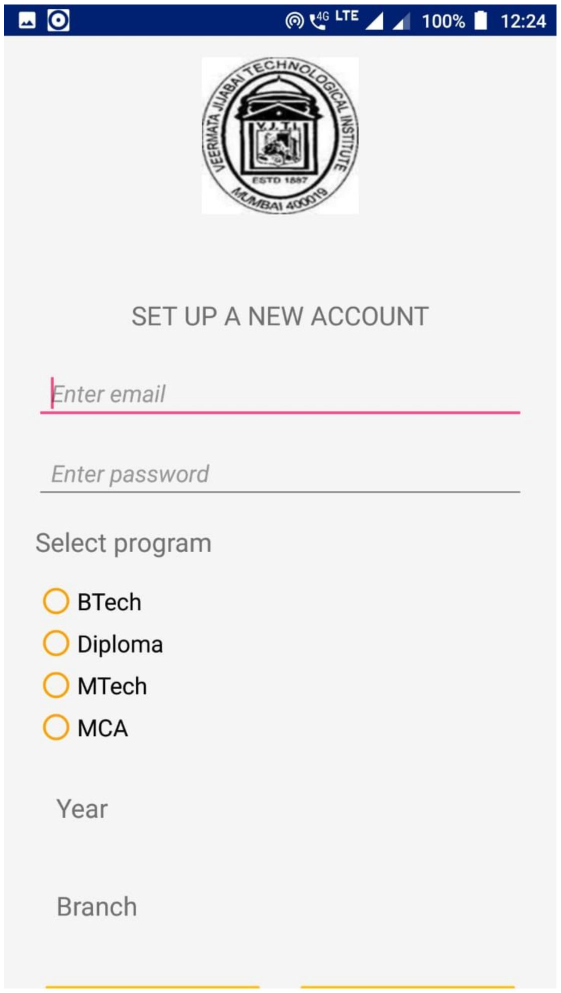
A LOOK AT THE APP

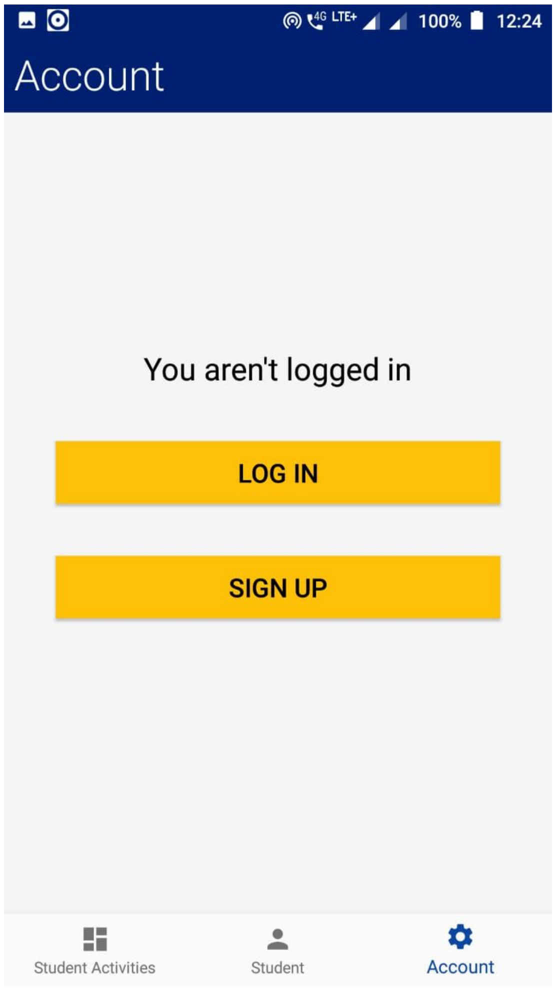




- Home Screen of V-Connect with Student Activities and Login button for committee heads.

- Login Screen for students of VJTI.


VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
V.J.T.I.
ESTD 1887
MUMBAI 400019
SET UP A NEW ACCOUNT
Enter email
Enter password
Select program
BTech
Diploma
MTech
MCA
Year
Branch


Account
You aren't logged in
LOG IN
SIGN UP
Student Activities
Student
Account

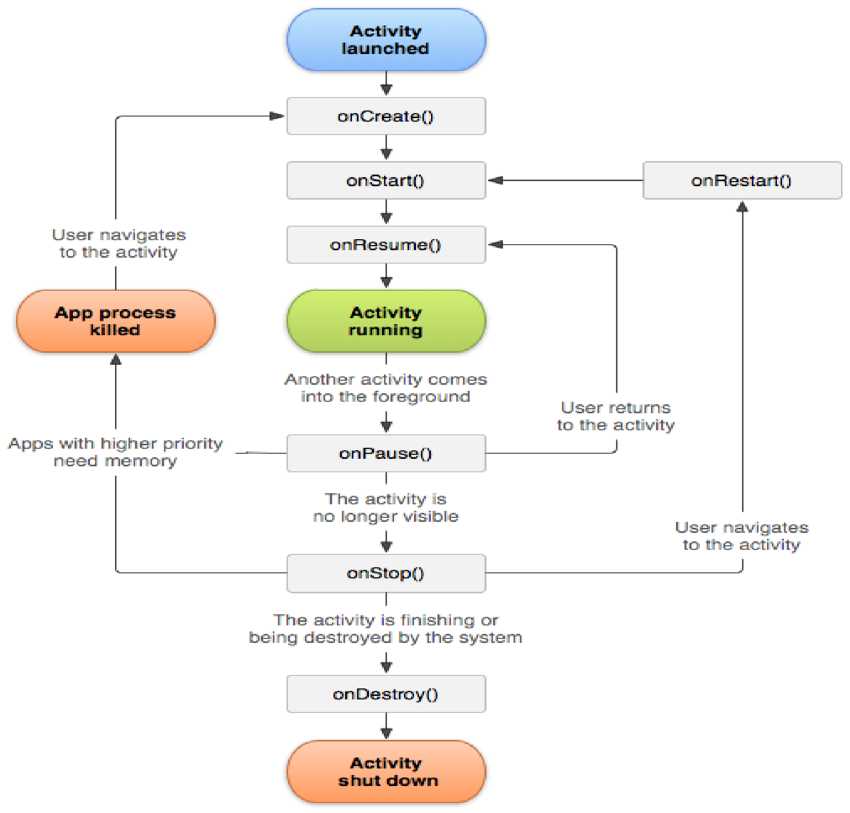
ANDROID BASICS

ACTIVITY

Unlike programming paradigms in which apps are launched with a `main()` method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle. In this way, the activity serves as the entry point for an app's interaction with the user. To use activities in your app, you must register information about them in the app's manifest, and you must manage activity lifecycles appropriately.

ACTIVITY LIFECYCLE

As the user begins to leave the activity, the system calls methods to dismantle the activity. In some cases, this dismantlement is only partial; the activity still resides in memory (such as when the user switches to another app), and can still come back to the foreground. If the user returns to that activity, the activity resumes from where the user left off.



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

ANDROID MANIFEST

Android Manifest is an .xml file that is used to provide specific information to the Android Build tools, the Android Operating System and Google Play.
Among many other things, the manifest file is required to declare the following:

- The app's package name, which usually matches your code's namespace. The Android build tools use this to determine the location of code entities when building your project. When packaging the app, the build tools replace this value with the application ID from the Gradle build files, which is used as the unique app identifier on the system and on Google Play.
- The components of the app, which includes all activities, services, broadcast receivers, and content providers. Each component must define basic properties such as the name of its Kotlin or Java class. It can also declare capabilities such as which device configurations it can handle, and intent filters that describe how the component can be started.
- The permissions that the app needs in order to access protected parts of the system or other apps. It also declares any permissions that other apps must have if they want to access content from this app.
- The hardware and software features the app requires, which affects which devices can install the app from Google Play.

In lieu of the project concerned, the permissions required from the user are :

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
```

The project requires internet access and to check whether the internet connection is presently available or not in a device is checked with the help of network state. Since the user requires downloading and uploading images, the permission of reading external storage and camera is required.

INTENT

An Intent is a simple message object that is used to communicate between android components such as activities, content providers, broadcast receivers and services. Intents are also used to transfer data between activities.

FEED ACTIVITY: CardView Using RecyclerView

In Android, a `CardView` is a convenient and stylistic element to represent information. For example, the individual news items one can find in Google Feed are represented as CardViews. `CardView` was introduced in Material Design in API level 21 (Android 5.0 i.e Lollipop). It is possible to display multiple CardViews in a list of CardViews, by combining `CardView` with `RecyclerView`. `RecyclerView` is an advanced and flexible version of `ListView` and `GridView`. It acts as a container that holds the current views being displayed. It has the benefit of being efficient and optimized, so this makes scrolling through all the elements smooth. The reason is explained in a very comprehensible manner in the Android Developers Documentation:

> The `RecyclerView` creates only as many view holders as are needed to display the on-screen portion of the dynamic content, plus a few extra. As the user scrolls through the list, the `RecyclerView` takes the off-screen views and rebinds them to the data which is scrolling onto the screen.

Every individual element in the `RecyclerView` is represented by a `RecyclerView.ViewHolder` object. These view holder objects are, in turn. managed by an adapter object, which can be created by extending `RecyclerView.Adapter`. The adapter is responsible for creating view holders as needed, and binds the data to be displayed to respective view holders.

The XML code below shows how the `RecyclerView` was implemented in the layout file for the Feed Activity:

```xml
<androidx.recyclerview.widget.RecyclerView
android:id="@+id/feed_recyclerview"
android:layout_width="match_parent"
android:layout_height="match_parent">
</androidx.recyclerview.widget.RecyclerView>
```

The `CardView` was implemented in XML as:

```xml
<androidx.cardview.widget.CardView
android:layout_width="match_parent"
android:layout_height="wrap_content" android:elevation="48dp"
app:cardCornerRadius="3dp">
<!--Views inside the CardView for title, description, date, et cetera
-->
</androidx.cardview.widget.CardView>
```

Representation of Posts and Class Notices

To conveniently store the data of individual posts and class notices internally a class titled `'Post'` was created. Its data fields (strings) consist of `title`, `description`, `date`, `id` (post key) and `image`(it helps in loading the image from a URL string). Its methods consist of setters and getters for all data fields, a default empty constructor and a constructor to construct a post object using passed title, description, date and image arguments.

```java
public Post(String title, String description, String image, String
date)
```

This manner of storing posts, which is a practical implementation of encapsulation, internally facilitates the following operations:

1. Adding, editing and deleting posts, and accessing them for the same.
2. Storing these posts in the Firebase Realtime Database. A great feature of Firebase Realtime Database is that it allows entire objects to be sent as data for storing them in the database. Hence there is no need to send attributes of a post individually to the online database.
3. Displaying these posts in individual CardViews using the view holders, since it is possible to pass all the attributes of a post in a single entity.
4. The image is stored in the database in the form of a URL directly accessed from Firebase Storage reference. The URL is converted into a string format to facilitate easy loading of images into image views via Picasso.

STORING USER DATA WITH SHARED PREFERENCES

In almost every application, it is necessary to store data related to the user on the user's device itself. For example, storing the username of the user, his/her birthday, his/her preferences, et cetera. This makes it convenient to remember details about the user, even after the user closes the app and can also give the user a sense of familiarity and comfort within the app.

For this app, it is necessary to store user data such as the credentials of the user, which are required for admins of the Student Activities. Also, for regular users, their program, year and branch also need to be stored so that content relevant to them can be shown (e.g. class notices). It is important to note that for these requirements we are storing only a small collection of information. For relatively less data, Android provides a great API called `SharedPreferences` that stores 'key-value' pairs in a file, called a shared preferences file, on the user's device. It can easily store primitive data types along with strings.

First, a `SharedPreferences` object is created in Home Activity. It is declared as public and static so that all activities can use the same `SharedPreferences` object.

```
public static SharedPreferences sharedpreferences;
```

Next, we get access to a shared preferences file (or if it doesn't exist, create one) called 'userCred' using the `getSharedPreferences()` method:

```
sharedPreferences = getSharedPreferences("userCred",
Context.MODE_PRIVATE);
```

This makes the `sharedPreferences` object a handle to the newly created 'userCred' shared preferences file. Internally, shared preferences files are XML files.

Now, for writing data to the 'userCred' file, we use an object of `SharedPreferences.Editor` class:

```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putBoolean("logged",true); // User logged in (Boolean)
editor.putString("login_id",loginId); // Store login ID (String) of
user
editor.commit(); // Save changes to the shared preferences file
```

The first parameter in `putBoolean()`, `putString()`, `putInt()`, et cetera methods is the 'key', while the second parameter is the 'value' we want to store.

Similarly, we can store program, year and branch of user as shown, in Student Signup Activity:

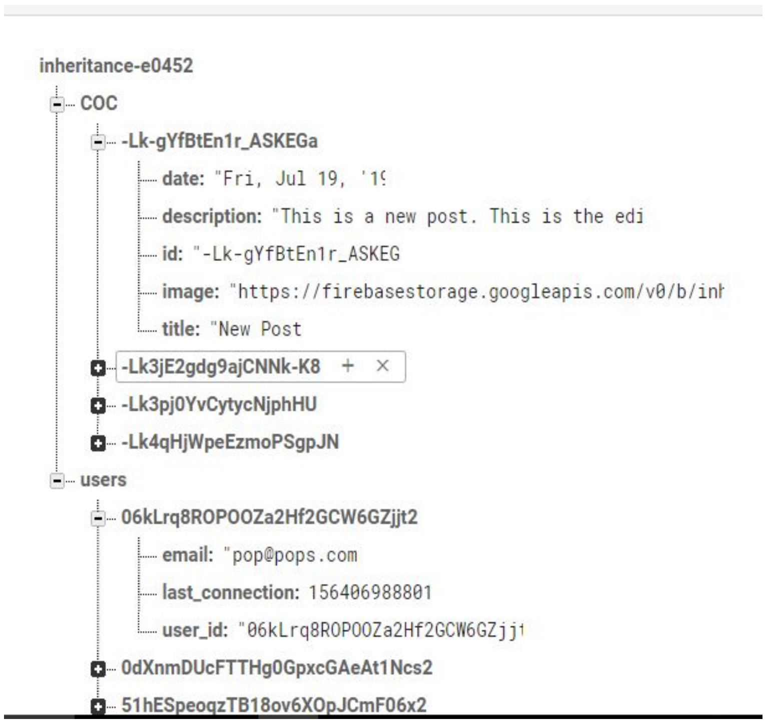
```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("program", program);
editor.putString("year", year);
editor.putString("branch", branch);
editor.commit();
```

For reading the values stored in the shared preferences file, we use `getString()`, `getBoolean()`, et cetera methods. In these methods, the first parameter is the 'key' of the data we want to retrieve, while the second parameter is the default value that the method should return in case the key passed to the method does not exist in the shared preferences file.

```
if(sharedPreferences.getBoolean("logged", false)){
      //User logged in
} else{
      //User not logged in
}
// Assigns login id in loginId if it exists, otherwise assigns null
loginId = sharedPreferences.getString("login_id",null);
```

FIREBASE REALTIME DATABASE

The Firebase Database is a cloud-hosted NoSQL Database. Data is stored here in the form of JSON and synchronized in realtime to every connected client. This database is used for storing and displaying posts as well as class notices.



```
inheritance-e0452
 ⊟-- COC
   ⊟-- -Lk-gYfBtEn1r_ASKEGa
        date: "Fri, Jul 19, '1
        description: "This is a new post. This is the edi
        id: "-Lk-gYfBtEn1r_ASKEG
        image: "https://firebasestorage.googleapis.com/v0/b/inh
        title: "New Post
   ⊞-- -Lk3jE2gdg9ajCNNk-K8  + ×
   ⊞-- -Lk3pj0YvCytycNjphHU
   ⊞-- -Lk4qHjWpeEzmoPSgpJN
 ⊟-- users
   ⊟-- 06kLrq8ROPOOZa2Hf2GCW6GZjjt2
        email: "pop@pops.com
        last_connection: 156406988801
        user_id: "06kLrq8ROPOOZa2Hf2GCW6GZjj
   ⊞-- 0dXnmDUcFTTHg0GpxcGAeAt1Ncs2
   ⊞-- 51hESpeoqzTB18ov6XOpJCmF06x2
```

The structure of our database is shown. It is a JSON tree with multiple nodes. Every committee is a parent node and every post added contains a unique ID, date, title and description.

The user is a node whose children are email, last_connection and user_id. The authorisation is used to create a primary key for handling events.

The implementation of Firebase Realtime Database in Android Studio is as follows :

In build.gradle file of app module :

```
implementation 'com.google.firebase:firebase-core:17.0.0'
implementation 'com.google.firebase:firebase-database:18.0.0'
implementation 'com.firebaseui:firebase-ui-database:0.4.0'
```

FIREBASE AUTHENTICATION

Firebase provides a full set of authentication options out-of-the-box. Firebase automatically stores users' credentials securely and redundantly (with replication and daily off-site backups). This separates sensitive user credentials from the application data, and lets developers focus on the user interface and app experience.

This project uses sign up with email and password. To create a new user with an email and a password, the following function is used:

```
auth = FirebaseAuth.getInstance();
auth.createUserWithEmailAndPassword(email, password);
```

This creates a new user with the given email address at the backend. The password is not available to the developer. To log in, the following code is used:

```
auth.signInWithEmailAndPassword(email, password);
```

An onCompleteListener can be added to facilitate the use of these methods. To check if the user is logged in,

```
if(auth.getUser == null)
//    User not logged in
else {
//   User logged in
userId = auth.getCurrentUser.getUid();
}
```

Firebase also has the feature of using other sign-in methods such as Google, GitHub, Facebook, Twitter and others. This makes the app very convenient and quick for the user.

BUGS AND ISSUES

1. Sometimes, user can see class notices only if he/she exits and reopens the app. This is due to a synchronization issue between Firebase Realtime Database and SharedPreferences.
2. Previous activities keep getting added onto the back stack, so the back button doesn't work as expected.

FUTURE WORK

1. Adding 3d map of college.
2. Integrating with college authorities and adding features like college notices, railway concession, etc.
3. Student to student communication through the app, with every student having a profile, like a social platform for VJTI
4. Forum for VJTI students.

BIBLIOGRAPHY

NAME	LINK
Android Developers	https://developer.android.com/index.html
StackOverflow	https://stackoverflow.com/
Abhi Android	https://abhiandroid.com/
Simplified Coding	https://www.simplifiedcoding.net/
Android Hive	https://www.androidhive.info/
Melardev	http://melardev.com/blog/category/android/

PROJECT LINK: https://github.com/ShubhankarKG/VJTI-VConnect

TEAM

AKSHAT SHAH

SHUBHANKAR GUPTA

SAHARSH JAIN

- INFORMATION TECHNOLOGY DEPARTMENT, VJTI 2018-22