
GitHub Username: https://github.com/whitefire00700

My Portfolio - Sylvester Francis

Description :

My Portfolio App in layman's terms is an App to showcase my CV. It is an intuitive way of attracting recruiters & contractors to my view my Resume. The Idea behind this app is to display my resume in a novel and unusual way to attract more recruiters to hire me

Intended User

The intended users are Recruiters and people who would likely want to hire me based on my profile

Features

- Displays my Location using Static maps Api
- Contact Me Page sends a mail to my mail account when the “Send Message” is pressed
- Uses Recyclerview to display a list of my projects and certificates and uses FireBase Real Time Database to store the data offline/online
- Uses Firebase Cloud messaging service to welcome users who’ve installed the app.

User Interface Mocks

Screen 1:

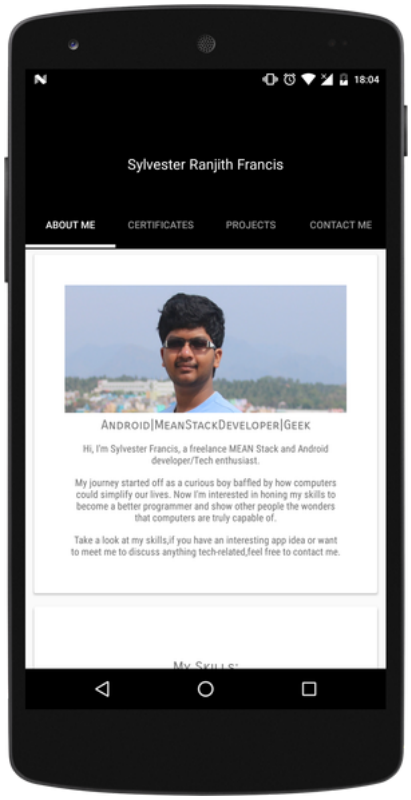


Fig 1.1.1: About Me Screen

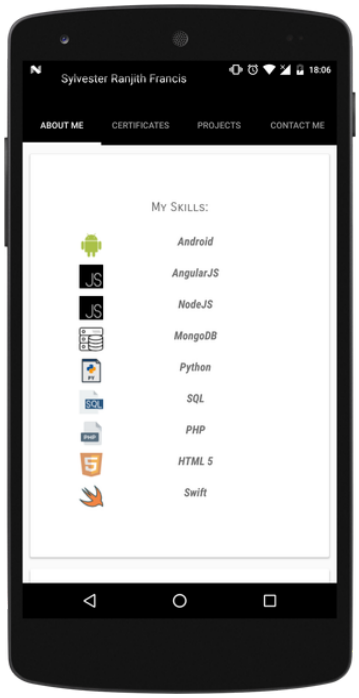


Fig 1.1.2: About Me Screen

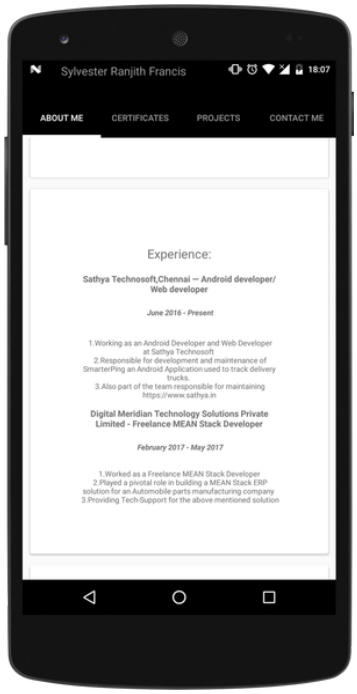


Fig 1.1.3: About Me Screen

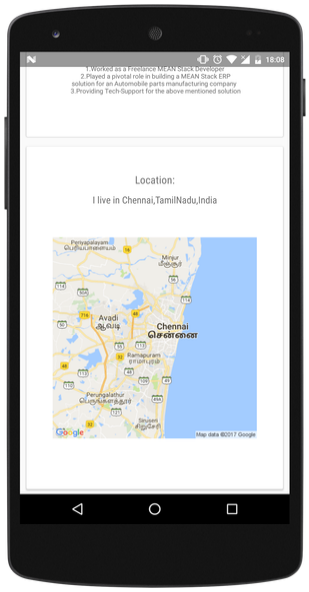


Fig 1.1.4: About Me Screen

DESCRIPTION – ABOUT ME SCREEN:

1. Fig 1.1.1 illustrates the first cardview of the ABOUT ME screen.
2. Fig 1.1.2 illustrates the second cardview of the ABOUT ME screen.
3. Fig 1.1.3 illustrates the third cardview of the ABOUT ME screen.
4. Fig 1.1.4 illustrates the fourth cardview of the ABOUT ME screen

The first cardview has an opening line for my portfolio app, It describes my current role and specialisation. The second cardview lists my skills and my strengths.The third cardview lists my experience and current employer details. The fourth cardview shows a map of my location using Google's Static Maps API.

Screen 2

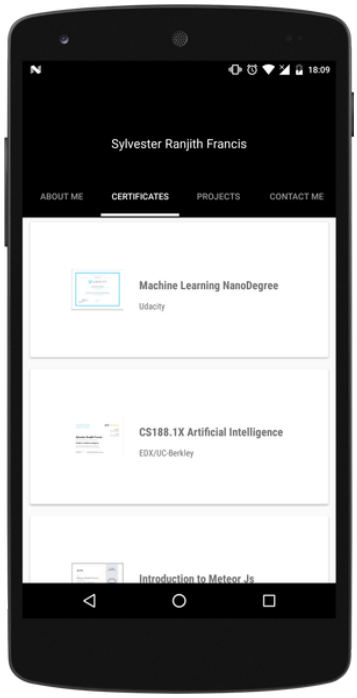


Fig 1.2.1: Certificate Screen

DESCRIPTION - Certificate SCREEN:

Fig 1.2.1 illustrates the recycler view of the Certificate screen.

This screen uses a combination of recyclerview and cardview to list all my certifications. This screen uses a custom adapter and a fragment. It uses Retrofit to get the data from a json document hosted online

Screen 3

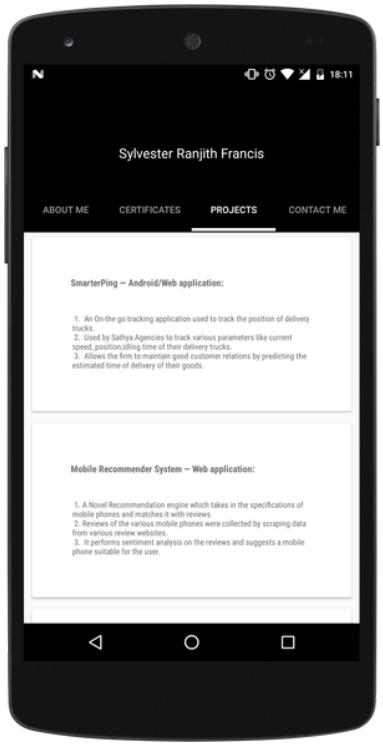


Fig 1.3.1: Project Screen

DESCRIPTION – Project SCREEN:

Fig 1.3.1 illustrates the recycler view of the Project screen.

This screen uses a combination of recyclerview and cardview to list all my projects. This screen uses a custom adapter and a fragment. It uses Retrofit to get the data from a json document hosted online.

Screen 4

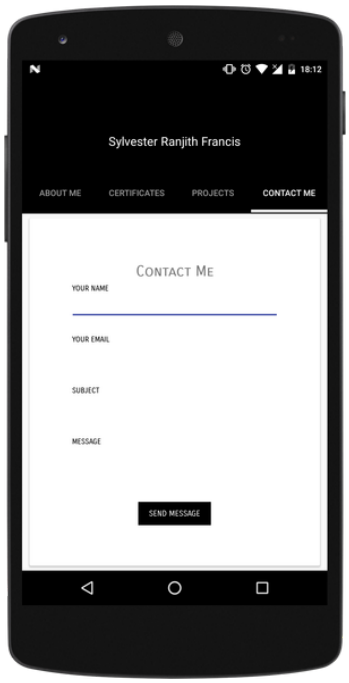


Fig 1.4.1: Contact Me Screen

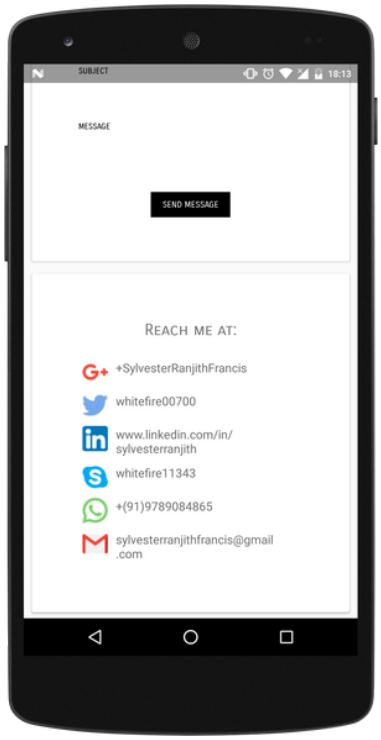


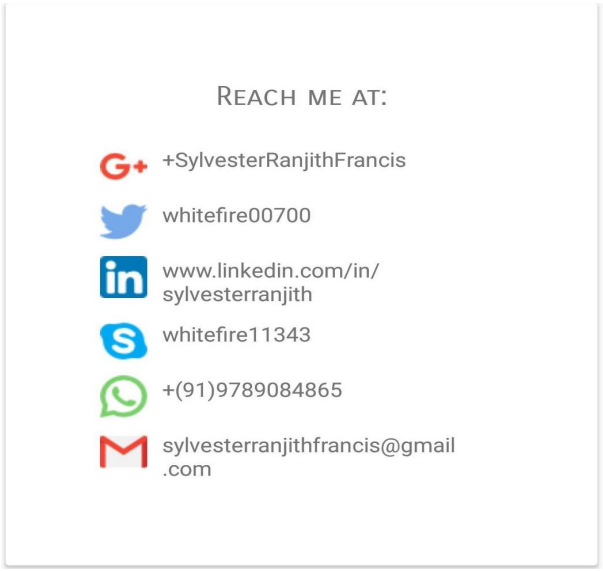
Fig 1.4.2: Contact Me Screen

DESCRIPTION – Contact ME SCREEN:

1. Fig 1.4.1 illustrates the first cardview of the Contact ME screen.
2. Fig 1.4.1 illustrates the second cardview of the Contact ME screen.

The Contact Me screen is used by a recruiter/contractor to contact me using the provided form.When the form is submitted it sends an entry into a mongodb database,notifying me of the sent message by also sending me an email. The second cardview of the Contact Me screen displays my contact information using which the recruiter can directly contact me.

Contact ME Widget:



The Contact Me widget is used by the recruiter to remind them of the quickest ways that they can reach me

Key Considerations

How will your app handle data persistence?

The app will use Firebase Real Time Database to query the json data hosted on the cloud.I would also use local-storage to save my Resume on the user's device when requested on click of a button

Describe any edge or corner cases in the UX.

Navigation within the app is mainly through swiping through the viewpager tabs present in the app. The only corner case that I will have to handle is maintain the responsiveness of the UI in all screen form-factors

Describe any libraries you'll be using and share your reasoning for including them.

Libraries Utilised to build the portfolio App:
Retrofit
ButterKnife
Picasso
Material ViewPager by florent 37 (https://github.com/florent37/MaterialViewPager)

Retrofit:

1. Retrofit is a type-safe REST client for Android (or just Java) developed by Square. The library provides a powerful framework for authenticating and interacting with APIs and sending network requests with OkHttp.

2. Retrofit is used in this app to Post Contact Form Data to a Rest API, Get the Static Map data from Google Static map api, Populate the individual Recycler Views with their respective data by sending a get request to the data present in a json format
3. It needs to pull or send data to/from a webservice or API only on a per request basis,to populate the certificate and project views. The app uses an AsyncTask on a background thread to achieve this to maintain the responsiveness of the UI/Main Thread

ButterKnife:

1. ButterKnife library is used for data binding of the UI and helps to reduce the UI boilerplate code.
2. Used extensively in binding data to their respective views

Picasso:

1. Picasso allows for hassle-free image loading in your application—often in one line of code!
2. Used for Caching the images in the Certificate Screen and About Me Screen efficiently

Material ViewPager by florent 37 :

1. A customisable Viewpager library used to reduce the hassle of designing the viewpager UI from scratch.
2. Used to load the beautiful viewpager layout of the app.

Describe how you will implement Google Play Services or other external services.

Google Analytics for Firebase:
Google Analytics for Firebase provides free, unlimited reporting in your app to measure user attribution and in-app activity such as screen views.I would use google analytics to figure out what parts of the app the recruiters viewed and make changes accordingly to the app

FireBase Cloud Messaging Service:

I would use firebase cloud messaging service to send messages from the console to recruiters who have installed the app informing them of my latest project or my current job experience.

Google Maps Static Api Service:

Google Maps static api service is used in the about me screen to inform the recruiters of where I'm currently posted or located

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

1. Create a new Android Studio Project with the title of the app as **"Sylvester Francis-CV"**
2. Select Empty Activity as the template of the MainActivity
3. Set the required gradle dependencies for the following

Florent 37 Material ViewPager :

```
compile 'com.github.florent37:materialviewpager:1.2.1'
```

```
//dependencies
compile 'com.flaviofaria:kenburnsview:1.0.7'
compile 'com.jpardogo.materialtabstrip:library:1.1.0'
compile 'com.github.bumptech.glide:glide:3.7.0'
```

RETROFIT :

```
compile 'com.squareup.retrofit2:retrofit:2.3.0'
```

PICASSO :

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

BUTTERKNIFE:

```
compile 'com.jakewharton:butterknife:8.7.0'
annotationProcessor 'com.jakewharton:butterknife-compiler:8.7.0'
```

4. Build gradle
5. Once Gradle has completed building the project setup is complete

Task 2: Implement UI for Each Activity and Fragment

Building UI for MainActivity:

1. Follow the steps in https://github.com/florent37/MaterialViewPager to set up a working viewpager
2. Edit the titles of the tabs to "**About Me**","**Certifications**","**Projects**","**Contact Me**"
3. Run the App to find the app with a ViewPager and 4 tabs present

Building UI for "**ABOUT ME**" screen:

1. Create a fragment layout file called "layout_aboutme"
2. Put in the respective layout elements like cardviews and customise the layout
3. Create a Custom Adapter for this view and call it "AboutMeAdapter" .
4. Set the adapter to the Recyclerview.
5. Run the app to find the About Me Screen Completed

Building UI for "**Contact Me**" screen:

1. Create a fragment layout file called "layout_contactme"
2. Put in the respective layout elements like cardviews and customise the layout
3. Create a Custom Adapter for this view and call it "ContactMeAdapter" .
4. Set the adapter to the Recyclerview.
5. Run the app to find the Contact Me Screen Completed

Building UI for “**Certificate**” screen:

1. Create a fragment layout file called “layout_certificates”
2. Put in the respective layout elements like cardviews and customise the layout
3. Create a Custom Adapter for this view and call it “CertificatesAdapter” .
4. Create setter and getter methods for the individual TextViews and Image Views
5. Set the adapter to the Recyclerview.
6. Run the App to find the basic version of Certificate screen completed

Building UI for “**Projects**” screen:

1. Create a fragment layout file called “layout_project”
2. Put in the respective layout elements like cardviews and customise the layout
3. Create a Custom Adapter for this view and call it “ProjectAdapter” .
4. Create setter and getter methods for the individual TextViews
5. Set the adapter to the Recyclerview.
6. Run the App to find the basic version of Project screen completed

Build Widget UI screen :

1.Design the layout for the Widget UI screen called “widget_info ”

2.Extend AppWidgetProvider. This class provides methods that are called during a widget lifecycle.

3.Provide the AppWidgetProviderInfo metadata. Essential information about the widget, such as minimum width and height and update frequency.

Task 3: Implement Retrofit API calls

Retrofit Api calls are to be implemented for the following screens:

1. About Me Screen (Get Call)
2. Project Screen (Get Call)
3. Certificate Screen (Get Call)
4. Contact Me Screen (Post Call)

Post Call is used to post the form data filled by the user of the app and also the necessary form validations are handled on the client

Task 4: Implement Picasso

Picasso library is used for image caching of the image resources present in the certificate screen and About Me screen

Task 5: Implement FireBase Cloud Messaging,Firebase Real Time Database

Implementation of FCM and Firebase Real Time Database are important for providing the user with an Onboarding Experience when the App is launched and also to send Server side messages to all the users of the app updating my current status to the users of the app

Task 6: Create Expresso Tests for the app :

Individual Tests are created for all the activities to test if they are functioning perfectly as designed. After this Task is Complete the App should be deployed to the Play Store.