**Rules of contribution & Documentation**

**Why am I building this app?**  
-> I want to build my own institution that stands in the world on its own. I need to finance that, to do that I want to expand in a previously untouched/neglected part of the world.  
I believe India’s population can be its biggest asset if utilised properly and that is what I want to do. I am also inspired by the huge number of people that are already working in the service industry.

**Rules of contribution –**

1. No half-hearted PRs will be accepted.
2. Every variable must be named according to guidelines.
3. Always remember -> Badly done is better than unfinished and nice.
4. Tell other people when they are stupid -> Nicely.
5. Add documentation for all Model Classes without fail.
6. Ask stupid questions.

**How to name variable?**

1. Classes follow -> CamelCase
2. Variables follow -> camelCase
3. NoSQL DB keys -> camelCase
4. Resource IDs -> @idNameOfResource
5. Do not be lazy -> make the name as long as you want
6. Activity classes -> ShoutOutLoudActivity.java
7. Model Classes -> Message.java/Advertisement.java/Mobile.java
8. Controller Classes -> WhatIsControlledController.java
9. Adapter Classes -> IronManSuitAdapter.java
10. Activity Layout -> activity\_thanos\_body\_modification.xml
11. Layout Resource -> layout\_resource\_name.xml

**Coding best Practices [Attention] –**

1. Every thing that goes in the Database HAS TO BE AN OBJECT ONLY or a getter() should exist for the value inserted.
2. Model Classes NEED to be present for all objects that will be put in a DB.
3. Every string used needs to be added to “strings.xml”.
4. String resources should be separated by the resource file names in a comment above it.
5. Every Image NEEDS to have a CONTENT DESCRIPTION.
6. Every Model Class that needs to output data to the user’s display NEEDS TO HAVE AN ADAPTER CLASS.
7. No variable should be PUBLIC.
8. If a function is used more than 2 times – create a utility & document it.
9. A total of 3 callback interfaces are available – USE THEM.
10. Every getter should return STRING or Boolean or Custom OBJECT only.
11. Don’t implement shit but, done >> unfinished.

**Rules for creating a Pull Request –**

1. Link your PR to the issue it concerns itself with.
2. If it is a major UI upgrade comment a Screenshot as well on the issue.
3. Your .gitignore needs to be an EXACT copy of the master branch.
4. Add out of current scope improvements to the To-do of the n+1 project.
5. Add @vikendu as reviewer.
6. If you are not @vikendu, DO NOT MERGE ANY PRs or MARK ANY ISSUES AS CLOSED.
7. Consider commenting something on the Issue if your branch went a little sideways while implementing.
8. Only the final version of the PR will be merged.
9. FORMAT – [Issue #xx] Branch name/something else.

**Roles that will be/are available –**

1. Developer - Implement features from 0 to 100, System Design, System Design issues, Database optimisation, be confident enough to close issues. [Available]
2. Developer Support – Resolve issues and TODOS. Research. Get reviewed by developers before closing issues. [Available]
3. Sales guy – Sell shit, approach investors. [Not Available]