**Standards Document**

**Project Naming**

* Project names will use Pascal Case i.e. TrustBuy. If necessary, use hyphens or underscores to separate words (not spaces).
* All Module names in the project shall start by TrustBuy followed by the date on which the code was written or updated. E.g. TrustApp\_27012017(ddmmyyyy).
* Major Modules of the project shall be in separate folders wherever possible and should be easily distinguished by clear Folder Name.
* While testing all related result files shall be named as TrustApp\_27012016\_Test and be put in a separate folder.

**Database Table Naming**

* Singular table name will be used.
* Singular column names will be used.
* To separate words in your table or column names, use Pascal Casing I.e. UserName, DateOfBirth.
* Name your P**rimary keys** using "[SingularOfTableName]ID" format. That is, whether your table name is Customer or Customers, the primary key should be CustomerID.
* Fields representing the same kind of data on different tables should be named the same. For instance do not have the column name as “Zip” in one table and “Zip\_Code” on another.

**Coding Standard**

|  |  |  |
| --- | --- | --- |
| **Identifier Type** | **Rule** | **Example** |
| **Classes** | Class names should be nouns. Avoid using abbreviations. | class Image; |
| **Methods** | Methods should be verbs with first letter lower case and each internal word capitalized. | run(); ,  runFast(); |
| **Variables** | Variable names shall be short and meaningful. Choice of variables shall be mnemonic – i.e. they should indicate the observer intent of its use. One character names can be used for temporary variables. | int i;  float myWidth ; |
| **Constants** | Variables declared as constants shall be all uppercase with words separated by underscore. | int MIN\_WIDTH = 4; |

If any identifier is used that is not mentioned in this document, then convention for that shall be agreed upon by all the team members.

**Documentation Comments**

**Block Comments**

Block comments may be used at the beginning of each file and before each method.

A block comment should be preceded by a blank line to set it apart from the rest of the code.

/\*

\* Here is a block comment.

\*/

**Single Line Comment**

Short comments can appear on a single line indented to the level of the code that follows. If a comment can't be written in a single line, it should follow the block comment format

**End-of-Line Comments**

The // comment delimiter can comment out a complete line or only a partial line. It shouldn't be used on consecutive multiple lines for text comments; however, it can be used in consecutive multiple lines for commenting out sections of code.

for commenting out sections of code. Examples of all three styles follow:

if (foo > 1) {

// Do a double-flip.

...

}

else {

return false; // Explain why here.

}

//if (bar > 1) {

//

// // Do a triple-flip.

// ...

//}

//else {

// return false;

**Documentation Comments**

Doc comments describe Java classes, interfaces, constructors, methods, and fields. Each doc comment is set inside the comment delimiters /\*\*...\*/, with one comment per class, interface, or member. This comment should appear just before the declaration:

/\*\*

\* The Example class provides ...

\*/

public class Example { ...

**Code Repository**

* All parts of source code will be contained in a private **Git Repository:**

https://github.com/Mkans/TrustBuy01262017

* Team members will work on their own separate branch from the master branch and only after approval of all team members after the review meetings their branch will be merged with the master branch.
* Under no circumstance, the project source files can be shared with anyone outside the team.