Indoor Mall Navigation - Coding Standards

Brute Force

May 2019

Contents

1 Introduction

The following document describes The coding standards that are to be practised during the development and implementation of the project 'Indoor Mall Navigation' by Brute Force for DVT.

This will provide a set of rules that will be used as guidelines and requirements for written programs.

2 Required and Optional Items

2.1 File Header

Each file is to contain a file header in the beginning of the code. The following is a summary of the file header items [1]

Item	Description
File Name	Name of file including it's extension and path to file
Version Number	This will be the file version number (this should be changed before every commit)
Author Name	This should be the name of the department working on the file
Project Name	The current program name
Organization	Name of the organization.
Copyright	Copyright information.
Related Documents	A list of the related documents
Update History	A list of updates which include author, name of person and reason for update
Functional Description	An overall description of the program (what it does)
Error Messages	A list of potential error messages that program may produce with a simple description of each message
Assumptions	Conditions that must be satisfied (these a conditions that affect the operation of the program
Constraints	Restrictions on the use of program

```
/**
 *  File Name: db.js (path: library/db.js)
 *  Version: 1.0
 *    Author: Brute Force - Database Management
 *    Project: Indoor Mall Navigation
 *    Organisation: DVT
 *  Copyright: (c) Copyright 2019 University of Pretoria
 *  Related Documents: Scan, Register, Login
 *
 *    Update History:
 *
 *  Date    Author     Changes
 *    -------------------------------------------
 *    10/04/2019  Thomas    Original
 *
 *    Functional Description: This program file establishes a database
      connection with the database
 *    Error Messages: Database Connection Failed
 *    Constraints: Can only be used to connect to database
 *    Assumptions: It is assumed that the database connect will be
      established
 *
 */
```

An example of a file header

2.2 Class Descriptions

Each class within the program should have a brief class description to ensure program readability and writeability across the developers involved. The required fields for each class description is the Purpose field. The other fields are a suggestion that is highly recommended in cases where the program will be written by multiple developers. [1]

Fields that would be found in the class description include:

- Purpose - A sentence stating the purpose of of the class
- Usage Instructions - A Simple description of how the class is to be used by other classes and methods (restrictions and constraints)
- Description of Methods- A simple description of the class methods which may include it's purpose, parameters, return type, and input and output files (exclude get and set methods)
- Description of Fields - a simple description of each field
- In-code Comments - comments within the class to help in the understanding and processing of the program (In-code comments will be found within the class and not necessarily outside of the class implementation code)

```
import { SQLite } from 'expo';

/**
 * Purpose    : This class is used to establish a database connection
     with SQLite
 *
 * Description : The class creates and establishes one connection to the
     database to allow following methods to alter the database using
     the already established connection.
 *
 * Usage       :
 */

export default class dbconnection{
   database: null;
   created: false;
   filled: false;
   open: false;

/**
 * Purpose    : A constructor that creates the database connection
 *
 * Description : Establishes connection with SQLite database
 */
   constructor(props)
   {
      this.database = SQLite.openDatabase("indoors");
      this.open = true;
      //console.log("opened");
      this.dropTable();
      this.buildTable();
      this.fillTable();
      //console.log(this.created);
   }
```

3 Coding Requirements and Conventions

3.1 Naming Conventions

Variables Variable names should be meaningful but not too long. Variable names must start with a lowercase letter and Camel Casing must be used if necessary. Every variable must be initialized prior to its first use.

```
var userLocation = new Location(this);
```

Classes, Subroutines, Objects, Functions and Methods The names of classes, subroutines, objects, functions and methods must describe its purpose, and start with a capital letter, Camel Casing must be used where necessary.

3.2 Formatting Conventions

Indentation Indentation is very important and should be consistent throughout the project. The body of a block should be indented and also the body of a structure like a loop should also be indented.

```
class App extends React.Component
{
    constructor(props)
    {
        super(props);
    }

    for(int i = 0; i < 10; i++)
    {
        LoopImplementation();
    }
}
```

Braces Braces should be on a new line to improve readability. Every method-/function should have braces, even if they contain one line.

```
// BAD!
if(shopFound)
    Navigate(shop);

// GOOD
if(shopFound)
{
    Navigate(shop);
}
```

3.3 In-code Comment Conventions

Inline Comments Inline comments should be used at a minimum, only when they are absolute necessary will inline comments be accepted because over-using them can cause the code to not be readable. Function Comments are encouraged over Inline Comments

```
for(int i = 0; i < 10; i++)
{
    LoopImplementation(); // Call the implementation for this loop
}
```

Function Comments Every function should have a comment block above it, Function Comments will include:

- A brief description of the function (What it does)
- The parameters it requires
- What it returns

```
/**
* Navigate User to shop
*
* @param shop - The shop to navigate to
* @returns null
*/
function Navigation(shop)
{
    // Navigate to shop
}
```

4 Rules and Responsibilities

4.1 Rules

These rules describe how the coding standards developed will be applied during the project

1. All programs should contain a file header with the following minimum requirements
    - File name
    - Copyright
    - Update History
    - Functional Description
2. Test case programs are subject to **Rule. 1**.

3. All coding conventions as stipulated in this document will be considered as coding requirements henceforth. This is subject to change throughout the development process to the discretion of the Project Manager and Organization.

4. Rules are subject to change throughout the development process to the discretion of the Project Manager, Organization and Developers assigned to the development and maintenance of such standards

4.2 Responsibilities

The following responsibilities have been designated to the following parties in order to achieve accountability and consistency throughout the development process.

1. The drafting and compiling of coding standards is assigned to Mr. Bandile Dlamini (Server Side Development) and Miss. Munyadziwa Tshisimba (Database Development and Management)

2. The educating of coding standards to all programmers is assigned to Mr. B Dlamini.

3. Follow up on program compliance with coding standards is assigned to Miss. M Tshisimba

4. Oversight is assigned to Mr. Thomas Honiball (Project Manager)

5 Conclusion

This document is to serve as a guideline. Developers/Programmers are to follow the minimum requirements as stipulated in the rules.

References

[1] David C. Kung. *Object-Oriented Software Engineering.* The Univrsity of Texas at Arlington. McGraw-Hill Companies, 2014. ISBN: 978259080791.