**Developer Documentation**

**Alejandria, Rizabelle**

**Panganiban, Elaine Jhon**

**Ju, Han Kyung**

**Table of Contents**

1. Coding Standards
   1. General Naming Conventions
   2. Files
   3. Statements
   4. Layout and Comments
2. Software Construction Strategy
3. **Coding Standards**
4. **General Naming Conventions**
   1. English – All names or words should be written in English since it is the language that is commonly used for development.

helloWorld

* 1. lowerCamelCase – conjoined names with mixed Cases starting with lower case so that each word can easily be read.

variableName

* 1. Underscore – conjoined names with an underscore after every word to understand the meaning of the word at a glance.

*method\_name()*

* 1. Misleading names – functions should be named after what they return and procedures after what they do to easily understand.

getName()

* 1. Similar Names – names that are reused and recycled are not desirable to avoid mixing them up.

product

products

Products

* 1. Acronyms – use of uppercase to the acronyms reduces the complexity of reading the acronym name and to stand out as it should.

HTMLlink

* 1. Abbreviation – bad abbreviation of words should be avoided as it slows down code reading. However, common abbreviation is allowed as it creates familiarity in which everyone can understand.

Valid: firstName, midInitial, lastName

Invalid: fName, mi, lName

1. **Files**
   1. Extension – files should have the extension used.

index.html

* 1. File name – file name should be match with the function of the file.

login.php

signup.php

* 1. Special characters – special characters must be avoided since these may cause problem for editors.

© ♥ ☆

* 1. Stye – uniform use of indention to be made obvious and speeds up code reading.

<html>

<head>

<title>Hello World</title>

</head>

<body>

<p>

This is a paragraph.

</p>

</body>

</html>

1. **Statements**
   1. Types – follow proper use of type for each variables.

float totalPrice;

int counter;

char midInitial;

* 1. Variables
     1. Initialization – Sometimes it is possible to initialize a variable to a valid value where it is declared. Also, it is possible that it should be left uninitialized rather than initialized.
     2. Arrays – arrays should be declared with their brackets next to the name;

int rose[];

double cheeseburger[];

* 1. Loops

1. for loop – loop control statements should not be included in the for() construction to make a clear distinction of what controls and what is contained in the loop.

totalPrice = 0;

for(i = 0; i < 100; i++)

totalPrice += value[i];

1. Initialization of loop – all variables, including loop variables, should be initialized from the start.
   1. Conditionals
      1. Complex conditionals – complex conditionals must be avoided so that the construction will be easier to read, debug, and maintain.
      2. if statement – the nominal case should be in the if-part and the exception in the else-part of an if statement.
      3. Separate line – the conditional statement should be on a separate line.

if (var == 0)

getLength();

* + 1. Executable statements – executable statements in conditionals must be avoided since it is very difficult to read.

float variableName = methodName();

if(variableName > 0){

// body

}

1. **Layout and Comments**
   1. Layout
      1. if-else statements

if (condition) {

statements;

}

else if (condition) {

statements;

}

else (condition) {

statements;

}

* + 1. for statement

for (initialization; condition; update) {

statements;

}

* + 1. while statement

while (condition) {

statements;

}

* + 1. do-while statement

do {

statements;

} while (condition);

* + 1. empty statement

if (condition)

statement;

while (condition)

statement;

for (initialization; condition; update)

statement;

* + - 1. Operators should be surrounded by a space character.
      2. Reserved words should be followed by a white space.
      3. Commas should be followed by a white space.
      4. Semicolons in for statements should be followed by a space character.

a = (b + c) \* d;

while (true) {

}

doSomething(a, b, c, d);

for (i = 0; i < 10; i++) {

}

* 1. Comments – comment out the details of what the method or variable does. The more detailed document, the better since it is useful when maximizing the use of comments.

/\* This is how the method works \*/

methodName(parameter1){

// body

}

1. **Software Construction Strategy**
2. **Programming Languages**
   1. Front-end
      1. HyperText Markup Language (HTML) – standard markup language used to create the web pages.
      2. Cascading Style Sheets – style sheet language written in a markup language used to describe the look and format of the documents.
      3. JavaScript – scripting language used for the web pages
   2. Back-end
      1. MySQL – database system used on the web
      2. PHP – server-side scripting language designed for developing the web

* 1. Server
     1. XAMPP – an open source used as web server solution
  2. Editor
     1. Brackets – modern text editor that makes it easy to design in the browser
     2. Sublime – slick user interface that makes the text editor appealing in coding.

1. **General Approach**

Logic vs. User Interface

The very first thing must do is to actually have an idea on what the structure of the program should take to accomplish the things needed. Also, it is important to focus on each detailed field and how do they interact which involves making ERD diagrams. A simple user interface to test a basic logic implementation at first is necessary simply to have an idea on the functionality of every fields. However, it tends to vary whether doing the logic or the user interface. There are times when the user interface is clearer than the logic or vice versa. But, in most cases the logic and user interface are conceptually similar. In a few words, our team decided to take the agile approach and work on small amounts of both in iterations.

Division of loads

Division of loads vary depending on how the structure of program looks like. Given the fact that Crossover Costumes is an e-commerce business, starting with the home page then per field (log in/sign up, contact us page, rental page, etc.) is most suitable.

Compliance of Code

Code compliance based on the coding standards provides a slick communication with the co-developers through the code. It purely shows teamwork plus it is much better to practice the coding standards of the team to avoid confusion from the members. So, it does not only help the developers to understand the codes but also the analysts, and testers.

Error Handling

Error handling can be very useful the fact of being aware what causes the error. It is important to take into consideration the error codes plus the default error messages to avoid blown error message with different formats especially to the user.

Programming

As a team, there should be no ownership of work. However, people work in different ways and different speeds. It can be quite frustrating working in pairs all the time, but somehow creates organizational stability. In general, individuals create better outputs by working alone. Also, working alone gets a chance working through a problem without someone interrupting.

In most cases, our team decided to work alone and use the version tool to update the source codes due to different schedules and priorities.

Version Control

A version control software such as GitHub allows us to track changes we make to our codes, and to easily backtrack these changes as well. Developer team upload the codes to GitHub, then check each other’s code and make changes when necessary. By using the fork tool in GitHub, it can create a copy of our repository and allows a member to freely experiment with changes without affecting the original files. Also, developer should have a simple integration-test to their code to check if it works and fixed it when necessary before uploading it to the GitHub.

Non-standard Language

The use of non-standard language is considered as an improper way of coding since following the coding standards is of extreme importance. Therefore, the team should comply with the coding standards talked about.

Test Cases

Preparing test cases before coding help the developers to understand how to design the code to make it convenient to use. Creating tests make the developers focus on what the function does. It is not the general practice but it is advisable in our team to write test cases before coding as it helps us learn to think outside the box.