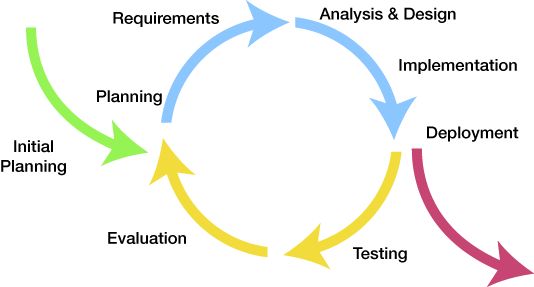
# SOFTWARE PROJECT MANAGEMENT　ソフトウェアプロジェクト管理

In this section, we will describe our organization and plan of project. Project team including team members must follow this section to complete assigned tasks and deadline. This also include introduction of our team and our project’s overview. Meeting minutes are also presented in this section. Moreover, we will describe risks of project and the way to communicate team members with each other.

## SOFTWARE DEVELOPMENT PROCESS　ソフトウェア開発のプロセス

The figure below will describe the information and product’s flow lifecycle process model.



*Figure 1.1 – Iterative software process model*

This is the process model which HOWE used to develop our system. There are many others process model that were widely applied in the world such as Waterfall, Spiral, V-model, etc, but we chose Iterative Software Process Model because of these following reasons:

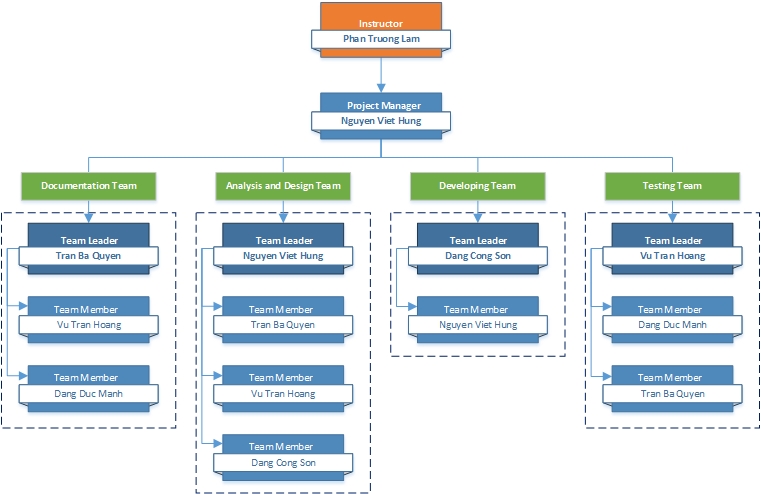
* The major requirements were defined clearly so by using this process model, we generate working software quickly and early during the sotfware life cycle.
* This model is more flexible – less costly to change scope and requirements.
* Easier to manage risk because risky pieces are identified and handle.
* When every single issue was discovered, it will be solved while the project is still on process.

After analyze, design, develope and test the main features of project, we will fix bugs, and test again. These actions will be repeated until the final product has been completed and ready to be packed.

The Iterative Process Model is suitable for team that doesn’t have much experience. It divides the development system task into smaller tasks which is easily review and rework.

## PROJECT ORGANIZATION プロジェクト組織

### Organizational Structure 組織構造



*Figure 1.2.1 – Organizational structure chart*

### Project Team プロジェクトチーム

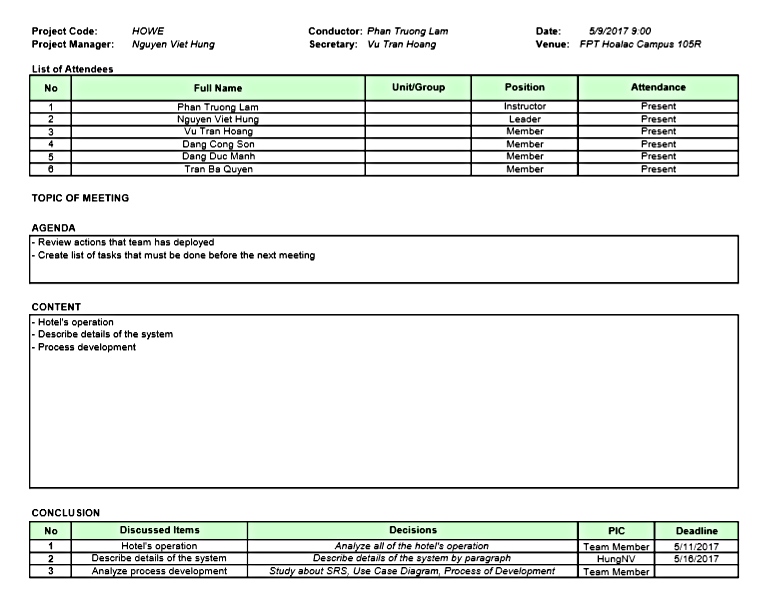
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Role | Responsibility | Full name | % Effort | Start date | End date |
| Instructor | - Provide templates & tools  - Review deliverables  - Review project status  - Resolve escalated issues  - Consultant | Phan Trường Lâm | 10% | 8th May, 2017 | 26th August, 2017 |
| PM | - Planning and defining scope  - Developing schedule  - Reviewing and tracking document  - Assign individual responsibility  - Assign task to team members | Nguyễn Việt Hưng | 100% | 8th May, 2017 | 26th August, 2017 |
| Analysis and Design Team | | | | | |
| Team Leader | - Reviewing and tracking design  - Identify main UI  - Draw prototype, architecture design  - Design Database | Nguyễn Việt Hưng | 100% |  |  |
| Team member #1 | - Draw prototype, architecture design | Trần Bá Quyền | 100% |  |  |
| Team member #2 | - Design database  - Design screen | Vũ Trần Hoàng | 100% |  |  |
| Team member #3 | - Design database  - Design screen | Đặng Công Sơn | 100% |  |  |
| Documentation Team | | | | | |
| Team Leader | - Reviewing and tracking requirement | Trần Bá Quyền | 100% |  |  |
| Team member #1 | - Create SRS Document  - Create Final Report | Vũ Trần Hoàng | 100% |  |  |
| Team member #2 | - Create SRS Document  - Create Final Report | Đặng Đức Mạnh | 100% |  |  |
| Development Team | | | | | |
| PTL | - Developing  - Solve issue of project | Đặng Công Sơn | 100% |  |  |
| Developer #1 | - Developing | Nguyễn Việt Hưng | 100% |  |  |
| QA and Testing Team | | | | | |
| Test Leader | - Responsible for test excution | Vũ Trần Hoàng | 100% |  |  |
| Tester #1 | - Testing | Trần Bá Quyền | 100% |  |  |
| Tester #2 | - Testing | Đặng Đức Mạnh | 100% |  |  |

*Table 1.2.2 – Project Team*

## PROJECT SCHEDULE　プロジェクトスケージュール

## MEETING MINUTES 打ち合わせ議事録

All meeting minutes will be written follow this template:



*Figure 1.4 – Meeting Minutes*

## PROJECT MILLSTONES　プロジェクトマイルストーン

|  |  |  |  |
| --- | --- | --- | --- |
| No | Milestone | Completion Date | Verification |
| 1 | Project Start | May 8th, 2017 | Instructor approval |
| 2 | Submit report 1 | May 22nd, 2017 | Instructor approval |
| 3 | Submit report 2 | June 5th, 2017 | Instructor approval |
| 4 | Submit report 3 | June 19th, 2017 | Instructor approval |
| 5 | Submit report 4 | July 24th, 2017 | Instructor approval |
| 6 | Submit report 5 | August 7th,2017 | Instructor approval |
| 7 | Submit report 6 | August 21st, 2017 | Instructor approval |
| 9 | Defend project |  | Instructor approval |

*Table 1.5 – Project Milestones*

## RESOURCES　リソース

### Human resource 人的リソース

* Team members
* Supervisor

### Non-human resource 機器やツール

* Equipment: Desktop, Laptop
* Building: Alpha & Beta Building of FPT University Hoalac Campus

## RISK MANAGEMENT　リスク管理

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Name | Root Cause | Probability | Avoidance plan | Contingency plan | Impact |
| 1 | Project team don’t meet the deadline on time. | Members lack of responsibility  Lack of management. | Medium | Make rules and penalties for member who didn’t meet the deadline. | Working overtime to complete tasks on time. | High |
| 2 | Requirement changed while project is being processed. | SRS is not provide all information about user requirement such as: lack of customer’s requirement, misunderstand customer’s requirement. | High | SRS should be reviewed carefully by PM and Supervisor.  Always do brainstorming carefully to design detail system. Hold all meeting and make minimize effort to change and improve design by requirement changes. | Team members will have meetings with supervisor to analyze the requirement changes, and make a specific action to resolve that problem. | High |
| 3 | Delivery does not meet the deadine. | Team members work late, lack of responsible causing not meet project schedule. | High | Assign tasks to suitable team members depend on their ability. | Team leader should regularly care about dealine and project schedule to inform members.  Break down task to smaller tasks and assign new tasks to team members. | Medium |
| 4 | Lack of knowledge about technology needs for project. | Depend on requirement, new technology should be apply in the project, which some team members haven’t used before. | Medium | Each team members must study to understand the technology, framework which is needed for project. | Technical leader need to support carefully for team members in group to increase required skill and knowledge.  Review complete task to make comment for team members. | Medium |
| 5 | Conflict among team members. | Team members don’t understand each other. | High | Plan some team building activities to improve the relationship between team members.  Clear role and responsibility for each member. | Have a meeting to resolve conflict problems. | High |
| 6 | Data or Source Code Lost | Delete Brand on GitHub.  Delete wrong source code. | Medium | Back up source code carefully.  Commit all small tasks after completed to GitHub. | Restore data from backed up data.  Restore source code from history versions. | High |

*Table 1.8 – Risk Management*

## COMMUNICATION MANAGEMENT コミュニケーション管理

### Communication between Team Members チームメンバーの間のコミュニケーション

* **Face-to-face meeting:** at least twice a week on Tuesday and Thursday. This is the fastest way of communication to solve big problems in the project and members can help out each other easily.
* **E-mail and message:** Email, Skype and Facebook are used for members to keep tracking other team member’s progress and team members can also help each other online.
* **Mobile Phone:** is using for emergency situation to directly contact to others.
* **Collaboration tool:** GitHub is used for document and source code management.

### Communication with Supervisor 指導教員とコミュニケーション

* **Face-to-face meeting:** Weekly on every Tuesday morning to make sure that supervisor can keep tracking of the team’s progress.
* **E-mail:** Gmail is the fastest way to get advice and document checking from supervisor.
* **Mobile phone:** is used to get time and place arranged for the meeting every week.

## CONFIGURATION MANAGEMENT PROCESS　コンフィグレーション管理のプロセス

### CI Identification and Naming Convention 構成アイテムと命名規則

|  |  |  |
| --- | --- | --- |
| No | Configuration Items | Naming Convention |
| Project Management | | |
| 1 | Project Plan |  |
| Requirement & Design | | |
| 2 | SRS |  |
| 3 | Architectural Design |  |
| 4 | Screen Design |  |
| 5 | Data Design |  |
| Source Code | | |
| 6 | Source Code |  |
| Support Document | | |
| 7 | User Manual |  |
| Test | | |
| 8 | Unit Test Plan |  |
| 9 | Integration Test Plan |  |
| 10 | System Test Plan |  |
| 11 | Unit Test Case |  |
| 12 | Integration Test Case |  |
| 13 | System Test Case |  |
| 14 | Test Data |  |
| 15 | Test Result |  |
| Process | | |
| 16 | Guideline |  |
| 17 | Template |  |
| 18 | Checklist |  |
| File Type | | |
| 19 | MS Word | \*.docx |
| 20 | MS Excel | \*.xlsx |
| 21 | MS PowerPoint | \*.pptx |
| 22 | MS Project Plan | \*.mpp |
| 23 | Images | \*.png or \*.jpg or \*.jpeg or \*.bmp or\*.gif |

*Table 1.9.1 – Naming Convetion*

### Project Infrastructure プロジェクトインフラストラクチャ

Below is the list of tools and infrastructure requirements needed for development environment.

#### Software ソフトウェア

|  |  |
| --- | --- |
| Operating system | Microsoft Windows 10 |
| Programming languages | PHP 5, HTML, CSS, Javascript, JQuery |
| Framework | Laravel 5.4, Bootstrap 3.3.4 |
| Version Control | Github 2.4 |
| IDEs | PHPStorm v2017.2, Notepad++, pgAdmin 4 |
| DBMS | PostgreSQL |
| Deployment tools |  |
| Communication tools | Skype, Facebook, Phone, Email. |
| UML tools | Astah Professional 7.1.0 |
| UI design tool | Balsamiq Mockups |
| Management tools | Microsoft Project Professional 2016 |
| Web server | Homestead 7 |
| Documentation | Microsoft Word 2016 |
| Microsoft Excel 2016 |
| Microsoft PowerPoint 2016 |

*Table 1.9.2.1 – Tools needed for project*

#### Hardware ハードウェア

* 5 laptops for developing, testing and creating documents with the configuration: 4GB Ram, 256GB of hard disk, Intel Core i5 processor.
* Internet network connection with minium speed 512kbit/s

#### Other infrastructure その他のインフラストラクチャ

* Rooms for meeting and working.
* Internet and mobile phone service for communication.

### Directory structure ディレクトリ構造

|  |  |  |
| --- | --- | --- |
| Main folder | Sub-folder | Purpose |
|  |  |  |
|  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |
|  |  |  |
|  |  |
|  |  |
|  |  |  |
|  |  |  |

*Table 1.9.3 – Directory structure*

### Coding convention コーディング規約

These following rules follow the standard rules of developing application using PHP. It refers to the coding convention on websites:

<https://www.codeigniter.com/user_guide/general/styleguide.html>

**File Format**

Files should be saved with Unicode (UTF-8) encoding. The BOM should not be used. Unlike UTF-16 and UTF-32, there’s no byte order to indicate in a UTF-8 encoded file, and the BOM can have a negative side effect in PHP of sending output, preventing the application from being able to set its own headers. Unix line endings should be used (LF).

Here is how to apply these settings in some of the more common text editors. Instructions for your text editor may vary; check your text editor’s documentation.

**PHP Closing Tag**

The PHP closing tag on a PHP document ?> is optional to the PHP parser. However, if used, any whitespace following the closing tag, whether introduced by the developer, user, or an FTP application, can cause unwanted output, PHP errors, or if the latter are suppressed, blank pages. For this reason, all PHP files MUST OMIT the PHP closing tag and end with a single empty line instead.

**File Naming**

Class files must be named in an Ucfirst-like manner, while any other file name (configurations, views, generic scripts, etc.) should be in all lowercase.

INCORRECT:

* somelibrary.php
* applicationConfig.php

CORRECT:

* Somelibrary.php
* applicationconfig.php

Furthermore, class file names should match the name of the class itself. For example, if you have a class named Myclass, then its filename must be Myclass.php.

**Class and Method Naming**

Class names should always start with an uppercase letter. Multiple words should be separated with an underscore, and not CamelCased.

INCORRECT:

* class superclass
* class SuperClass

CORRECT:

* class Super\_class {}

Class methods should be entirely lowercased and named to clearly indicate their function, preferably including a verb. Try to avoid overly long and verbose names. Multiple words should be separated with an underscore.

INCORRECT:

* function fileproperties() // not descriptive and needs underscore separator
* function fileProperties() // not descriptive and uses CamelCase
* function getfileproperties() // Better! But still missing underscore separator
* function getFileProperties() // uses CamelCase

CORRECT:

* function get\_file\_properties() // descriptive, underscore separator, and all lowercase letters

**Variable Names**

The guidelines for variable naming are very similar to those used for class methods. Variables should contain only lowercase letters, use underscore separators, and be reasonably named to indicate their purpose and contents. Very short, non-word variables should only be used as iterators in for() loops.

INCORRECT:

* $j = 'foo'; // single letter variables should only be used in for() loops
* $Str // contains uppercase letters
* $bufferedText // uses CamelCasing, and could be shortened without losing semantic meaning
* $groupid // multiple words, needs underscore separator
* $name\_of\_last\_city\_used // too long

CORRECT:

* for ($j = 0; $j < 10; $j++)
* $str
* $buffer
* $group\_id
* $last\_city

**Commenting**

In general, code should be commented prolifically. It not only helps describe the flow and intent of the code for less experienced programmers, but can prove invaluable when returning to your own code months down the line. There is not a required format for comments, but the following are recommended.

Comment code block:

/\*

\* @author Author Name

\* @link http://example.com

\*/

Use single line comments within code, leaving a blank line between large comment blocks and code:

* // break up the string by newlines
* // $parts = explode("\n", $str);

**Constants**

Constants follow the same guidelines as do variables, except constants should always be fully uppercase. Always use CodeIgniter constants when appropriate, i.e. SLASH, LD, RD, PATH\_CACHE, etc.

INCORRECT:

* myConstant // missing underscore separator and not fully uppercase
* N // no single-letter constants
* S\_C\_VER // not descriptive
* $str = str\_replace('{foo}', 'bar', $str); // should use LD and RD constants

CORRECT:

* MY\_CONSTANT
* NEWLINE
* SUPER\_CLASS\_VERSION
* $str = str\_replace(LD.'foo'.RD, 'bar', $str);