**Report No.2: Project Management Plan (PMP)**

**2.1. Problem Definition**

2.1.1. Name of this Capstone Project

* Official name: Scrap collector
* Vietnamese name: Ứng dụng hỗ trợ thu mua ve chai
* Abbreviation: SCH

2.1.2. Problem Abstract

As current Vietnamese scrap collectors have to waste a lot of time to go around and find who has scrap to sell. This traditional way is ineffective, for example, the collectors don’t know exactly where to go, so they just go around and ask loudly for scrap, which takes their time, their force and harms to their health.

This project is concern about implementing information technologies and mapping direction algorithms into scrap collectors’ daily work. We call it Scrap Collector Helper System (SCH). SCH provides the homeowner an easy way to post the scrap they want to sell with detail information and the recommended price. Besides that, SCH also provides the collector a feature to find out near-by scrap, book scrap and contact with the homeowner. Moreover, SCH helps the collector to save time and force by providing the optimized route to collect all booked scrap. Finally, SCH gives homeowner chances to review the collector and those collector with good review will have priority in their work.

2.1.3. Project Overview

2.1.3.1. The Current Situation

Below are the problems encountered in this project:

* Limit of time: our team has only 4 members and we have to get requirements, complete document, develop product and do the test in just 14 weeks.
* New technique and framework: when applying new framework into capstone project, all members of the team need an amount of time to get used to new techniques.
* Lack of User Interface and User Experience design skill: all team members are studying JS for major, no one has studied about UI and UC designing before.
* Misunderstanding in business logic: the business login of scrap collecting is quite strange with us, and all requirements are given by the mentor, who is not a scrap collector. Although we have observed many scrap collectors, there are still some misunderstood logic.
* Absence of team member: team members can have a conflict in meeting schedule because of sick, class timetable or work schedule, etc…

2.1.3.2. The Proposed Systems

According to the technology researches, we found out that the Dijkstra's Algorithm (Dijkstra's Shortest Path First algorithm) is very capable of resolving the problem of collecting scrap. Because of its popularity and the high applicability, we decided to choose Dijkstra's Algorithm to use in application.

We also build a high available web server to maintain the main system to work 24/7 to make sure that if mobile applications have the need of accessing to the server to get or update the information there will always be available.

We assign responsibility in vertical to make sure if any member in this problem cannot continue to work in our team there will be the least harmful to the project processes.

Our system includes 2 main subsystems:

- An online website for the admins to help them manage user account, category and scrap post.

- A mobile application for homeowner and collector to sell and buy scrap.

2.1.3.2.1 Web application

Web application is common tool to help admin manage user, category and scrap post. This application is used by administrator team and do followings:

* + Manage user account
  + Manage scrap category
  + Manage posted scrap

2.1.3.2.2 Mobile application

Mobile application is a tool to help homeowner and collector to sell and buy scrap. Mobile application provide following features:

* For homeowner:
  + Post Scrap for sale
  + Contact the collector
  + Manage scrap
  + Negotiate scrap price
  + Update information
* For collector:
  + Find near-by scrap
  + Book scrap
  + Contact the homeowner
  + Schedule scrap
  + Negotiate price

2.1.3.3. Boundaries of the System

The system can:

* Allow Admin to manage user accounts .
* Allow Admin to manage list category of scrap.
* Allow Admin to manage scrap posted by homeowner.
* Allow Homeowner to post scrap for sale.
* Allow Homeowner to negotiate scrap price.
* Allow Homeowner to contact the collector.
* Allow Homeowner to review the collector.
* Allow Collector to find near-by scrap.
* Allow Collector to negotiate scrap price.
* Allow Collector to schedule a pickup.
* Allow Collector to contact the homeowner.

The system cannot:

* Help collector make payment online.

2.1.3.4. Development Environment

2.1.3.4.1 Hardware requirements

For server

|  |  |  |
| --- | --- | --- |
| **Hardware** | **Minimum Requirement** | **Recommended** |
| **Internet Connection** | Cable, Wi-Fi (8 Mbps) | Cable, Wi-Fi (50 Mbps or more) |
| **Operation System** | XP, Vista, 7, 10, Window Server 2008 | 10, Window server 2008 |
| **Computer Processor** | Intel® Xeon ® 1.4GHz | Intel® Xeon ® Quad Core (12M Cache, 2.50 GHz) |
| **Computer memory** | 4GB RAM | 32 GB RAM or more |
| **Storage space** | 1GB | 8GB or more |

Table 1 - Hardware Requirements for Server

For PC

|  |  |  |
| --- | --- | --- |
| **PC** | **Minimum Requirements** | **Recommended** |
| **Internet Connection** | Cable, Wi-Fi (4 Mbps) | Cable, Wi-Fi (8 Mbps) |
| **Operating System** | Window 7 | Window 7 or more. |
| **Computer Processor** | Intel® Core i3 1.4GHz | Intel® Core i5 2.50GHz |
| **Computer Memory** | 1GB RAM | 2GB RAM or more |
| **Web Browser** | Chromes (v42 or higher) | Chrome latest stable version |

Table 2 - Hardware requirement for PC

2.1.3.4.1 Software Requirements

|  |  |  |
| --- | --- | --- |
| **Software** | **Name / Version** | **Description** |
| **Operation System** | Windows Server 2014 | Operating system and platform for development |
| **Environment** | Java EE 8.0, Node v10, npm v6, react-native v0.61 | Specification for developing web application |
| **Modeling tool** | StarUML | Used to design diagram |
| **IDE** | Visual Studio Code 1.27.2, A5M2 2.12.1 (SQL) | Programming tools |
| **DBMS** |  | Used to create & manage the database for system |
| **Source control** | GitHub, GitHub Desktop | Used for source control |
| **Web browser** | Chrome 69 or above | Testing browser |

Table 3 - Software Requirements

**2.2. Project organization**

2.2.1. System Process Model

We have decided to apply the Modified Waterfall Model in developing project. Below are reasons why we choose this kind of model:

The waterfall model has many advantages, one of its advantage is to help developers understand well requirements from customer. Besides that, it is simple and easy to understand and use.

The most popular disadvantage of the traditional waterfall model is the low ability to adapt with changes, once an application is in the lower stage, it is very difficult to go back and change something that was not well-thought out in the previous stage. However, in the modified waterfall model, all the phrases are the same, but each one can overlap each other to maximize the use of benefits and minimize losses from disadvantages.

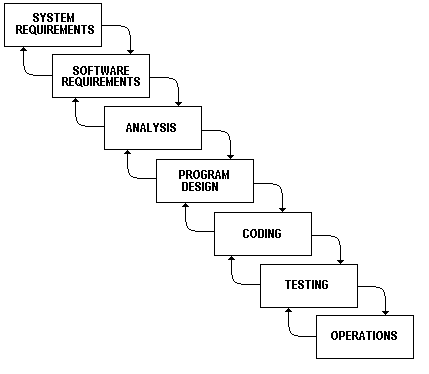


Figure 1 – Modified Waterfall Model

***Reference:*** *Page 63, chapter 5, The Software Life Cycle, MEDICAL DEVICE SOFTWARE VERIFICATION, VALIDATION, AND COMPLIANCE, by David A.Vogel.*

2.2.2. Roles and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full name** | **Role in Group** | **Responsibilities** |
| **1** | Lại Đức Hùng | Product Owner | * Specify user requirement * Control the development process * Give out technique and business analysis support |
| **2** | Nguyễn Duy Đạt | Scrum Master | * Managing process * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create test plan * Coding * Testing * Arrange Meeting * Risk Management |
| **3** | Tiết Lê Bảo Khánh | Scrum team member | * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create test plan * Coding * Testing |
| **4** | Nguyễn Hoàng Anh Trung | Scrum team member | * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create test plan * Coding * Testing |
| **5** | Nguyễn Hoàng Nhân | Scrum team member | * Clarifying requirements * Prepare documents * GUI Design * Create test plan * Coding * Testing |

Table 4 - Roles and responsibilities Details

2.2.3. Tools and Techniques

|  |  |
| --- | --- |
| Tool/Technique | Name |
| Front-end | HTML, CSS, JavaScript, jQuery, React, AngularJS |
| Back-end | JavaEE, SpringBoot framework, JPA, Hibernate |
| IDE | Visual Studio Code |
| DBMS | Firebase, PostgreSQL |
| Source Control | GitHub, GitHub Desktop |
| Modelling tool | StarUML |

Table 5 - Tools and techniques

**2.3. Project management plan**

2.3.1. Software development life cycle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Description** | **Deliverables** | **Resource needed** | **Dependencies and**  **Constrains** | **Risks** |
| **Requirement**  **Analysis** | - Research about requirements from scrap collectors.  -Identify requirements for the system in common. | -Software requirement specification.  -Introduction of proposed application.  -Project Task scheme  - Prototypes | 20 man- days | N/A | - Uncertain of project's scope  - Lack of experience in collect requirement |
| **Design** | - Architecture design of all the system needed.  - UI, UX design  - Design ERD  - Design Component diagram | - Base source code for back-end and front-end  - Base UI  - ERD  - Component diagram | 30 man- days | All design depend on  “Requirement  Analysis” | - Lack of experience in design UI, UX. |
| **Implementation** | - Build system core functions.  - Build real-time APIs and other APIs.  - Implement mobile application.  - Implement website application for admin. | - Main user’s functions on the mobile application for Pet Owner. | 50 man-days | All implementation depends on “Design”. | - Human misunderstandings.  - Lack of knowledge and experience. |
| **Testing** | - Test approach  - Test plan  - Test case | - Test document | 20 man-days | Depend on  “Implementation” | - Lack of experience in testing  - Unaccounted for test case |
| **Maintenance** | - Deploy on sever and mobile | -Installation guide.  - User Manual | 10 man- days | Depend on  “Testing”  and “Implementation” | - Require experience in maintenance. |

Table: Software development life cycle

2.3.2. Phase Detail

2.3.2.1 Phase 1: Requirements Analysis

| **Task** | **Description** | **Author** |
| --- | --- | --- |
| **1. Collect requirements** | Find which application currently provide similar service. |  |
| **2. Identify and clarify the main functions** | Define core application functions. |  |
| **3. Create system** **introduction** | Fulfill Introduction Report. |  |
| **4. Software project**  **management plan** | Prepare Project Management scheme. |  |
| **5. Prototype** | Build a prototype of proposed system. |  |
| **6. SRS** | Create SRS document. |  |

2.3.2.2 Phase 2: Design

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Database design | Based on ERD.  Based on other needs to design. |  |
| 2. Technology research | Study React-native, Redux, React-redux framework and study AngularJs framework. |  |
| 3. Design document | Create software design document. |  |

2.3.2.3 Phase 3: Implementation

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. API functions** | Implement APIs for Scrap Collector application. |  |
| **2. Web Admin** | Implement a web page for admin. |  |
| **3. Mobile functions** | Implement mobile application. |  |
| **4. Unit testing** | Write test cases and test for web functions. |  |
| Write test cases and testing for mobile functions. |  |

2.3.2.4 Phase 4: Testing

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Integration testing** | Write test case and test systems. |  |
| **2. Testing document** | Write test case documents. |  |
| **3. Acceptance testing** | Let customer test real product to ensure the product meet user’s requirements. |  |

2.3.2.5 Phase 5: Operation and Maintenance

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Installation guide** | Write installation guide |  |
| **2. User’s Manual** | Write user manual |  |

2.3.3. All Meeting Minutes

All sprint meeting minutes could be found [here](../Meeting_minus).

**2.4. Convention Rules**

**Summary**:

* **Naming Convention**:
* Variable names should be short yet meaningful. The choice of a variable name should be designed to indicate to the casual observer the intent of its use.
* Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.
* **Indentation**:
* One declaration per line is recommended since it encourages commenting.
* In absolutely no case should variables and functions be declared on the same line.
* Do not put different types on the same line.
* **Declarations Convention:**
  + One declaration per line is recommended since it encourages commenting.
  + Using Java Code Convention from:

<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

**2.5. Other material (if any)**