**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

**DOCUMENT REPORT**

Capstone Project Document

**VN Habit Tracker**

|  |  |
| --- | --- |
| **Group 3** | |
| **Group members** | Lưu Thành Đạt - SE61124  Nguyễn Quang Tuyến - SE62069  Nguyễn Hữu Thắng - SE62447  Phạm Thanh Tùng - SE61628 |
| **Supervisor** | Lại Đức Hùng |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | VHT |

- Ho Chi Minh city, September 14th 2018

Table of Contents

[List of tables 4](#_Toc530436935)

[List of Figure 5](#_Toc530436936)

[Definitions, Acronyms, and Abbreviations 6](#_Toc530436937)

[**A.** **Introduction** 7](#_Toc530436938)

[1. Project Information 7](#_Toc530436939)

[2. Introduction 7](#_Toc530436940)

[3. Current Situation 7](#_Toc530436941)

[4. Problem Definition 7](#_Toc530436942)

[5. Proposed Solution 7](#_Toc530436943)

[5.1 Feature functions 8](#_Toc530436944)

[5.2 Advantages and disadvantages 8](#_Toc530436945)

[6. Functional Requirements 9](#_Toc530436946)

[7. Role and Responsibility 9](#_Toc530436947)

[**B.** **Software Project Management Plan** 10](#_Toc530436948)

[1. Problem Definition 10](#_Toc530436949)

[1.1. Name of this Capstone Project 10](#_Toc530436950)

[1.2. Problem Abstract 10](#_Toc530436951)

[1.3. Project Overview 10](#_Toc530436952)

[2. Project organization 13](#_Toc530436953)

[2.1. Software process model 13](#_Toc530436956)

[2.2. Roles and responsibilities 14](#_Toc530436957)

[2.3. Tools and Techniques 15](#_Toc530436958)

[3. Project Management Plan 16](#_Toc530436959)

[3.1. Software development life cycle 16](#_Toc530436961)

[3.2. Phase Detail 17](#_Toc530436962)

[3.3. All Meeting Minutes 19](#_Toc530436963)

[4. Coding Convention 19](#_Toc530436964)

[**C.** **Software Requirement Specification** 21](#_Toc530436965)

[1. User Requirement Specification 21](#_Toc530436966)

[1.1 Guest Requirement 21](#_Toc530436967)

[1.2 User Requirement 21](#_Toc530436968)

[1.3 Admin Requirement 22](#_Toc530436969)

[2. System Requirement Specification 22](#_Toc530436970)

[2.1 External Interface Requirement 22](#_Toc530436971)

[2.2 System Overview Use Case 23](#_Toc530436972)

[3. Software System Attribute 24](#_Toc530436973)

[3.1. Usability 24](#_Toc530436974)

[3.2. Reliability 24](#_Toc530436975)

[3.3. Availability 24](#_Toc530436976)

[3.4. Security 24](#_Toc530436977)

[3.5. Maintainability 24](#_Toc530436978)

[3.6. Portability 24](#_Toc530436979)

[3.7. Performance 24](#_Toc530436980)

[4. Conceptual Diagram 25](#_Toc530436981)

[**D.** **Software Design Description** 26](#_Toc530436982)

[1. Design Overview 26](#_Toc530436983)

[2. Web architecture design 27](#_Toc530436984)

[2.1. Web Application architecture description 27](#_Toc530436985)

[2.2. Mobile Application architecture description 28](#_Toc530436986)

[3. Component Diagram 29](#_Toc530436987)

[4. DetailedDescription 32](#_Toc530436988)

[4.1. Class diagram 32](#_Toc530436989)

[4.2. Class Diagram Explanation 33](#_Toc530436990)

[4.3. Interaction Diagram 36](#_Toc530436991)

[4.4. Activity diagram 38](#_Toc530436992)

[5. Database Design 39](#_Toc530436993)

[5.1. Entity Relationship Diagram 39](#_Toc530436999)

[5.2. Data dictionary 39](#_Toc530437000)

[6. Database Relationship Diagram 41](#_Toc530437001)

[6.1. Physical diagram 41](#_Toc530437008)

[6.2. Data dictionary 42](#_Toc530437009)

[7. Algorithms 42](#_Toc530437010)

[7.1. Recommend the best habits for users by linear programming algorithm 42](#_Toc530437011)

# List of tables

[Table 1: Roles and Responsibilities 8](#_Toc530436897)

[Table 2 : Hardware Requirement for Server 11](#_Toc530436898)

[Table 3 : Hardware Requirement for Mobile 11](#_Toc530436899)

[Table 4 : Software requirements 12](#_Toc530436900)

[Table 5 : Roles and responsibilities 14](#_Toc530436901)

[Table 6: Tools List 14](#_Toc530436902)

[Table 7: Technique List 14](#_Toc530436903)

[Table 8: Software Development Life Cycle Detail 15](#_Toc530436904)

[Table 9: Phase 1: Infrastructure 16](#_Toc530436905)

[Table 10: Phase 3: Web service 17](#_Toc530436906)

[Table 11: Phase 4: Mobile app 18](#_Toc530436907)

[Table 12 - Conceptual Diagram Dictionary 25](#_Toc530436908)

[Table 13 - Web Components Dictionary 29](#_Toc530436909)

[Table 14 - Mobile Components Dictionary 30](#_Toc530436910)

[Table 15 - Class Diagram 32](#_Toc530436911)

[Table 16 - App User 32](#_Toc530436912)

[Table 17 - User 33](#_Toc530436913)

[Table 18 - Admin 33](#_Toc530436914)

[Table 19 - Habit 34](#_Toc530436915)

[Table 20 - Feedback 34](#_Toc530436916)

[Table 21 – Reminder 34](#_Toc530436917)

[Table 22 - Tracking 35](#_Toc530436918)

[Table 23 - Entity Relationship Diagram Dictionary 39](#_Toc530436919)

[Table 24 - Physical Diagram Dictionary 41](#_Toc530436920)

# List of Figure

[Figure 1 : Waterfall model 13](#_Toc530436921)

[Figure 2 – Class Diagram System Overview Use Case 22](#_Toc530436922)

[Figure 3 - Conceptual Diagram 24](#_Toc530436923)

[Figure 4 - Web Application Architecture Description 26](#_Toc530436924)

[Figure 5 - Web Component Diagram 28](#_Toc530436925)

[Figure 6 - Mobile Component Diagram 29](#_Toc530436926)

[Figure 7 - Class Diagram 31](#_Toc530436927)

[Figure 8 - Sequence Diagram for add new habit 35](#_Toc530436928)

[Figure 9 - Sequence Diagram for edit habit 36](#_Toc530436929)

[Figure 10 - Sequence Diagram for delete habit 36](#_Toc530436930)

[Figure 11 - Activity Diagram for habit 37](#_Toc530436931)

[Figure 12 – Entity Relationship Diagram 38](#_Toc530436932)

[Figure 13 - Physical Diagram 40](#_Toc530436933)

[Figure 14 - Linear Programming Algorithm 44](#_Toc530436934)

# Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Name** | **Definition** |
| VHT | VN Habit Tracker |
|  |  |

# **Introduction**

## Project Information

* Project name: **VN Habit Tracker**
* Abbreviation: **VHT**
* Product Type: **Web app & Mobile app**
* Start Date: **September 10th 2018**

## Introduction

Habits are routine behaviors done on a regular basis. They are recurrent and often unconscious patterns of behavior and are acquired through frequent repetition. Many of these are unconscious as we don’t even realize, we are doing them.

Good habits are hard to develop when we use the typical tactic of trying to pump up our motivation to overcome our psychological resistance. Research has shown that motivation is an ineffective factor or creating lasting behavior change. Fortunately, we have also will build an excellent alternative, which is called VHT. It will let us tunnel right through the resistance barrier.

First thing is to track habits! You can also track recurring behavior that you want to keep under control. This tracker can be made for daily use, so you get a very clear overview of all your good and bad habits. By keeping a habit tracker, you are more conscious of your behaviors. You know actually what you are doing, more importantly, what not, what you should do. Without this tracker it can be very easy to believe you are indeed doing everything you want, but in the meantime, secretly, you are not. The tracker is a tool to transform your goals into good habit.

## Current Situation

* Everyday our lives are governed by habit. These habits are the little routines and small ways of doing things.
* Most people are unaware or are only slightly aware that they’re doing them.
* VHT helps you to set goals, keep track of your habits and tackle your bad habits.

## Problem Definition

* **Disadvantages:**
  + Vietnamese users do not have the official application for tracking habit.
  + Customer easily give up on using.

## Proposed Solution

Our proposed solution is to build an application called “VN Habit Tracker” to resolve the current problems. The application collects (in real-time) all habits from customer. Moreover, it specifically helps them to track their habits daily, weekly and monthly. This application will motivate them by showing the notification and reminder which depend on their setting. Customer can check their currently process. We also design the app which has statistics that helps customer to see their efforts.

VHT system is an app application with following functions:

### Feature functions

* + Plan habits with an easy-to-use interface.
  + Habits fit into a number of different schedules, including specific times or days.
  + View stats in a number of different ways, with different data point.
  + Reminders.
  + Make sure no habit is forgotten with daily reminders.
  + Data Export.
  + Export your data to Excel or Numbers (CSV) and chart your progress.
  + Interactive Notifications.
  + Complete or snooze a habit right from the notification banner.
  + Weekly Targets.
  + Set how many days a week you would like to complete a habit.
  + Notes.
  + Write or dictate notes to capture all details of a habit. Great for exercise routines and daily journals.
  + Skip Functionality.
  + Going on vacation or taking a day off? Skip habits without breaking your chain.
  + Report.

### Advantages and disadvantages

These are advantages and disadvantages of current situation:

* **Advantages:**
  + - Customer will be easy to understand.
    - Friendly interface.
    - Motivate customer to change their bad habits.
    - VHT saves times and space for customer.
  + **Disadvantages:**
    - Advertising strategy difficulty of application.

## Functional Requirements

Function requirements of the system are listed as below:

* Add habits:
  + Choose habit: add a bad habit you want to reform or build a good one to improve.
  + Name habit: name that habit.
  + Choose period: choose daily, weekly, monthly.
  + Set goal: set up the goal you want to archive.
  + Choose group: set that habit belongs to which group that you want.
  + Adjust reminder: set time for the app to remind you.
  + Write description: write something that can motivate you.
  + Show statistics: show the statistics to display the chart in the present or in the past.
* Adjust setting:
* Set goal.
* Set reminder.
* Adjust filter: display the result in the form that customer want.
  + Choose goal period.
  + Choose goal type.
  + Choose goal values.
* Adjust chosen habit:
  + Edit chosen habits.

## Role and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
| 1 | Lại Đức Hùng | Project Supervisor | Supervisor | Hungld5@fe.edu.vn |
| 2 | Lưu Thành Đạt | Developer | Leader | Datltse61124@fpt.edu.vn |
| 3 | Nguyễn Quang Tuyến | Developer | Member | Tuyennq62069@fpt.edu.vn |
| 4 | Nguyễn Hữu Thắng | Developer | Member | Thangnhse62447@fpt.edu.vn |
| 5 | Phạm Thanh Tùng | Developer | Member | Tungptse61628@fpt.edu.vn |

Table 1: Roles and Responsibilities

# **Software Project Management Plan**

## Problem Definition

### Name of this Capstone Project

* **Official name**: VN Habit Tracker
* **Vietnamese name**: Ứng dụng theo dõi, cải tiến thói quen cho người Việt.
* **Abbreviation**: VHT

### Problem Abstract

Currently, in Viet Nam, the majority of people use smartphones every day. However, there is not a specific application for Vietnamese people to create habits, manage daily habits and track habits. The existing tracking habit applications are from foreign developer and thus may not really applicable for Vietnamese users. So we have to find the best convenient way to make our application simplest and easy to use for anyone.

To solve this particular problem, we decided to provide an application that helps users do that*.* VHT will help users to create habits, group habits, and prompt users to implement those habits. Create a goal for the user to try to achieve that habit over time and manage the statistics of the user’s system habits.

### Project Overview

#### Current Situation

Below are the problems encountered in this project:

* **User Habit:** people do not have a habit to set habit and manage habit by an application.
* **User:** easily give up when they depressed, unmotivated to continue a habit.
* Currently there is no application dedicated to Vietnamese.
* **Server crash**: The user's data will be lost when the server crashes. Because all data is stored on the server.
* **Interface**: user easy-to-use interface in VHT’s application.

#### The Proposed System

According to the survey of Vietnamese users, we find that most Vietnamese do not have a habit of scheduling, do not schedule what week they will do. So our solution now is to create an application specifically for Vietnamese users to improve and create their habits.

We build a system that is always maintained so that users can log in and use it on their mobile applications anytime, anywhere.

To solve the problem of creating habits and improving the habits of Vietnamese users, we need to know the Vietnamese people, what makes them do not create their daily routine.

Our system consists of two main systems: website and mobile applications. On the website, administrators manage users and statistical habits of Vietnamese. For mobile applications, we allow users to create and manage their habits, and we also receive feedback from Vietnamese users.

##### *Web Application*

Web application consists of three main parts:

* For administrator:
  + Manage User.
  + Manage Feedbacks
  + View statistics

Besides, website application also provides an API interface for two mobile applications to retrieve, update data from mobile applications.

##### *Mobile Application*

The mobile applications included functions as below:

* User:
  + Manage habit.
  + Group habit.
  + Habits statistic.
  + Suggest Habit.
  + Tracking habit.
  + Feedback.
  + View top habits.
  + Manage profile.
  + Manage settings.
  + Export data.

#### Boundaries of the System

* This application is built on the habits of Vietnamese. Our main target is helps them to set goals, keep track of habits and tackle bad habits.
* VN Habit Tracker which deployed this application must set up devices, includes:
  + Data export to CSV files.
  + Interactive notifications.
* The completed product includes:
  + Website application.
  + Android mobile application.

#### Future Plans

* With further research and development, the system can apply the following features:
  + Extend and create community using VHT.
  + Develop new features to set and remind users to create a habit: real time, remind by location..v..v..
  + Combined with smart watch and external devices to manage and message users.
  + Habit follow group.
  + Habit challenges.
  + Statistical data by location.
  + Recommend habits according to user information.
  + Bigdata & AI.

##### *Hardware requirements*

* **For web application server**

|  |  |  |
| --- | --- | --- |
| Windows | Minimum Requirements | Recommended |
| Internet Connection | Cable, Wi-Fi (4 Mbps) | Cable, Wi-Fi (8 Mbps) |
| Operating System | Window Server 2008 R2 | Window Server 2012 R2 |
| Computer Processor | Intel® Xeon ® 1.4GHz | Intel® Xeon ® Quad Core |
| Computer Memory | 2GB of RAM | 4GB of RAM or more |

Table 2 : Hardware Requirement for Server

* **For Mobile**

|  |  |  |
| --- | --- | --- |
| Android | Minimum | Recommended |
| Internet Connection | Wi-Fi or 3G (1 Mbps) | Wi-Fi or 3G (8 Mbps) |
| Operating System | Android 4.4.2 | Android 6.0.0 |
| Mobile Processor | Cortex-A7 Dual-Core 1.3GHz | Cortex-A7 Dual-Core 1.3GHz |
| Mobile Memory | 1GB of RAM | 2GB of RAM or more |

Table 3 : Hardware Requirement for Mobile

##### *Software requirements*

|  |  |  |
| --- | --- | --- |
| Software | Name / Version | Description |
| Operating system | Window Server 2012 R2 | Operating system and platform for development |
| Environment | Java | Specification for developing web application |
|  |  |  |
| IDE | IntelliJ IDEA, Android Studio v3.2.1 | Used for implement website and Android Mobile Application. |
| Design Model tool | Star UML v2.5.1 | Used for creating modal and diagrams. |
| DBMS | SQLite & MySQL | Used to create & manage the database for system |
| Document storage | Trello | Used for storing document |
| Store and manage source code | Git Hub | Used to store all source code |
| Web browser | Chrome or above | Testing browser |

Table 4 : Software requirements

## Project organization



### Software process model

The software process model used in developing the VHT project is based on the Water Fall model. We choose this model because of the following reason:

* The features of this product is strongly connected, therefore, a solid architect and clear requirement need to be setup in order to un the project smoothly.
* The requirements are not supposed to be changed rapidly.



Figure 1 : Waterfall model

Reference: Page 30, chapter 2, Software process model, SOFTWARE ENGINEERING 9th Edition, by Ian Sommerville.

### Roles and responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| No | Full name | Role in Group | Responsibilities |
| 1 | Lại Đức Hùng | Supervisor, Project Manager | * Specify user requirements * Control the development process * Give out technique and business analysis support |
| 2 | Lưu Thành Đạt | Team leader, B.A, Developer, Tester | * Managing process * Designing database * Clarifying requirements * Prepare documents * GUI design * Create test plan * Coding * Testing |
| 3 | Nguyễn Hữu Thắng | Team member,  B.A, Developer,  Tester | * Designing database * Clarifying requirements * Prepare documents * GUI design * Create test plan * Coding * Testing |
|  | Phạm Thanh Tùng | Team member,  B.A, Developer,  Tester | * Designing database * Clarifying requirements * Prepare documents * GUI design * Create test plan * Coding |
| 4 | Nguyễn Quang Tuyến | Team member,  B.A, Developer,  Tester | * Clarifying requirements * Prepare documents * Create test plan * Coding * Testing |

Table 5 : Roles and responsibilities

### Tools and Techniques

|  |  |
| --- | --- |
| Tool | Name / version |
| Web server | IIS |
| Development tool | IntelliJ IDEA, NetBean, Android Studio |
| DBMS | MySQL |
| Source control | Github |
| Modeling tool | Star UML v5.0.1 |
| Document tool | Microsoft Word 2016 |

Table 6: Tools List

|  |  |
| --- | --- |
| Technique | Name / version |
| Frontend | HTML5, CSS, JavaScript, WordPress, Sketch. |
| Backend | Java, Spring Boot, Hibernate, PHP, Android, Retrofit. |

Table 7: Technique List

## Project Management Plan



### Software development life cycle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Description | Deliverables | Resource needed | Dependencies and Constrains | Risks |
| Infrastructure | - Identify and clarify overall requirements.  - Determine the system architecture.  - Build infrastructure for the project. | - Database design.  - System main structure. | 20 man-days | N/A | - Unclear project scope.  - Lack of member share of understand. |
| API services | - Identify requirements for mobile app.  - Build required API for mobile app. | - API for mobile app. | 20 man-days | - Depends on “Database Design”. | - Lack of experience. |
| Mobile apps | - Design the mobile UI  - Build mobile apps for end users and emulator. | - Complete Android Apps. | 60 man-days | - Depends on “API services”. | - Lack of experience.  - Lack of Habit Behavior knowledge |
| Web App | - Implements all web app modules.  - Design the web UI.  - Build the web app. | - Complete Web Apps. | 20 man-days | - Depends on “API services”. | - Lack of experience. |

Table 8: Software Development Life Cycle Detail

### Phase Detail

#### Phase 1: Infrastructure

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Assessment | - Determine requirements.  - Determine requirements for System and API. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 2. Selection | - Determine system architecture: Java Web & Android.  - Determine software design pattern: Repository & Service.  - Determined all core functions. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 3. Development | - Create the main structure of project. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 4. Review | - Review all completed works and presentation.  - Create sprint backlog. | * DatLT * TungPT |

Table 9: Phase 1: Infrastructure

#### Phase 3: Web service

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Assessment | - Determine requirements for Web service. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 2. Selection | - Determine all functions according to requirements of Web service. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 3. Development | - Create API for mobile app based on functions on the web app. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 4. Review | - Review all completed works and presentation.  - Create sprint backlog. | * DatLT * ThangNH * TuyenNQ * TungPT |

Table 10: Phase 3: Web service

#### Phase 4: Mobile app

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Assessment | - Determine requirements for System and Mobile app. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 2. Selection | - Determine all functions according to requirements of Mobile app. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 3. Development | - Implement all the functions based on the designed UI and the provided API. | * DatLT * ThangNH * TuyenNQ * TungPT |
| 4. Review | - Review all completed works and presentation.  - Create sprint backlog. | * DatLT * TungPT |

Table 11: Phase 4: Mobile app

### All Meeting Minutes

Meeting minutes are contained in folder “Meeting minutes”.

### Coding Convention

This “Coding Convention” is using for develop project in both Java web admin and Android application:

* Naming Conventions:
* Folders (Java web & Android): folders name are always unique and written in lowercase.

Ex: customer, servlet, sample…

* Packages (Android application): packages/folders name are always unique and written in lowercase.
* Classes/Interfaces: classes or interfaces should be noun, name using Pascal case, in mixed case with the first letter of each internal word capitalized.

Ex: User, ImageSession, CustomerDAO…

* Methods: methods should be verb, name using Camel case, in mixed case with the first letter lower case, with the first of each internal word capitalized.

Ex: run (), crawlNew (), getSearchResults () …

* Variables: name using Camel case, in mixed case with the first letter lower case, with the first of each internal word capitalized. Variable names should not start with underscore \_ or dollar sign $ characters, even though both are allowed.

Ex: count, myVariable, isValidated…

* Constants: constants should be all uppercase with words separated by underscores.

Ex: MAX\_VALUE, GET\_SEARCH\_RESULT

* Comment:
* All source files should begin with a comment that lists the class name, description, date, author:

/\*

\* Class name

\*

\* Description

\*

\* Date

\*

\* Author

\*/

* All methods should begin with a comment that lists the method name, description, date, author:

/\*

\* Method name

\*

\* Description

\*

\* Date

\*

\* Author

\*/

* Comment code is 2 slashes “//”
* Comment to explain code need 4 slashes “////”
* Block comment used to provide description of files, methods, structures and algorithms.
* Indentation: Four spaces should be used as the unit of indentation
* Declarations: one declaration per line. Do not put different types on the same line

Ex: int minValue;

# **Software Requirement Specification**

* + - 1. User Requirement Specification
  1. Guest Requirement

Guest is a person who doesn’t have access to the system. Guest can use some functions in the system. To use all functions, guest must login. These are some functions that guest can use:

* Login.
* Register.
  1. User Requirement

User is a guest who logged into the system with user’s role. There are some functions that user can use:

* Manage habit:
* View habit
* Delete habit
* Edit habit
* Add habit
* Manage group:
* Add group
* Delete group
* Edit group
* Manage settings:
* Sort habit
* Set reminders
* Export data
* Feedback and contact
* View statistics
* Sugguest habit
* View top habit
* Manage profile
* Log out
  1. Admin Requirement

Admin is the person who manages the system, a super user of system. There is some functions admin can use:

* Manage users
* Reset password
* View statistics
* Manage feedbacks
* View feedbacks
* Reply feedback

1. System Requirement Specification
   1. External Interface Requirement
      1. *User Interface*

* General requirement for graphic user interface is the GUI should be simple, clear, intuitive, and reminiscent.
* The interface design is an iterated process includes design, sketching and user assessment.
  + 1. *Hardware Interface*
* Smartphone, desktop, laptop or tablet has the ability to connect to the internet.
* The system will use the standard hardware and data communications resources of a standard computer.
  + 1. *Software Interface*
* Run with Chromes (v60 or higher), Firefox (v46.1 or higher)
* The screen must bigger than 1024x768.
* Mobile application: Android studio (version 3.1.4), Genymotion (version 2.12.2).
  + 1. *Communication Protocol*
* Using HTTP/HTTPS protocol.
  1. System Overview Use Case

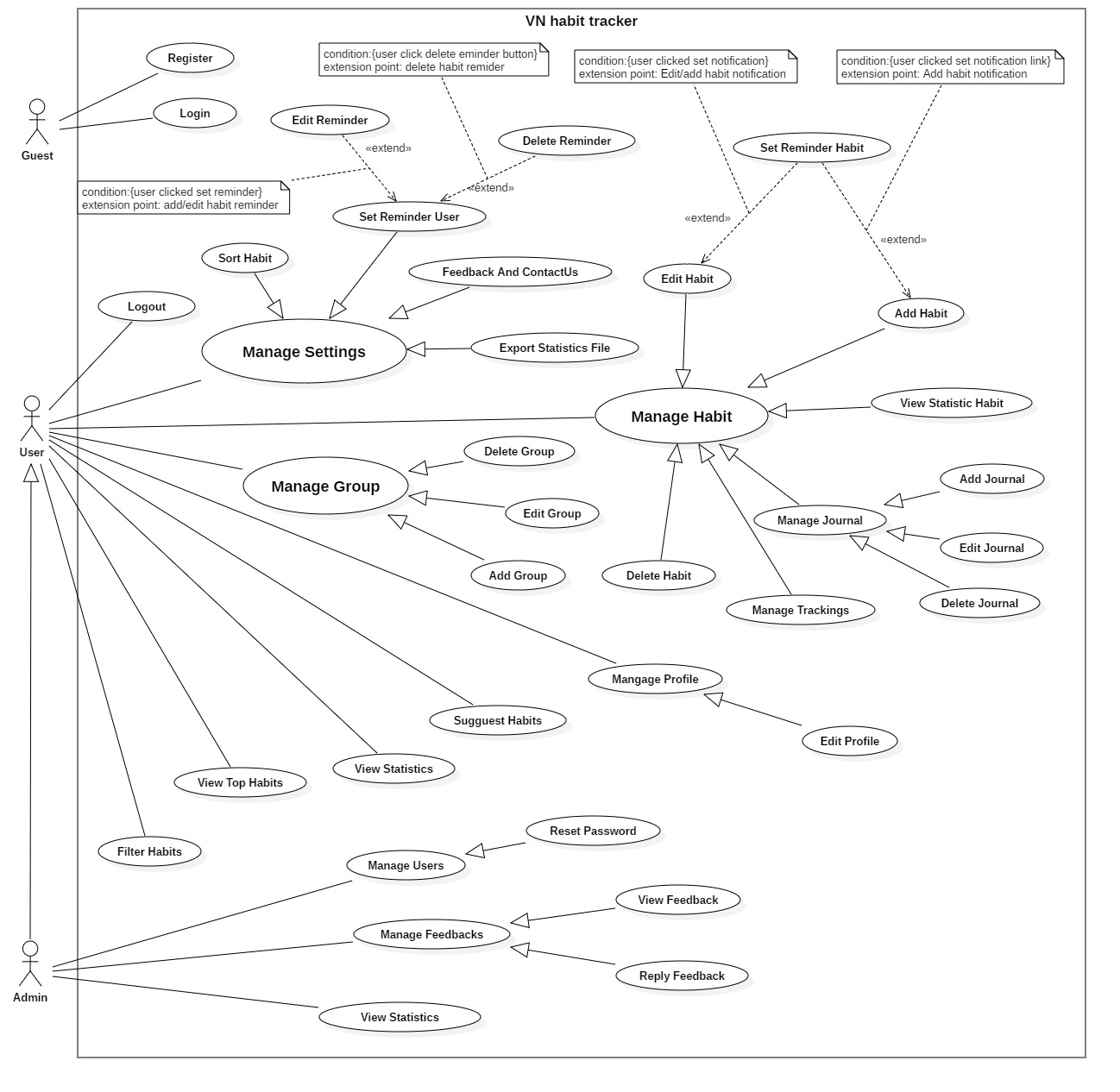


Figure 2 – Class Diagram System Overview Use Case

1. Software System Attribute
   1. Usability

* Provide a convenient way to interact with system.
* Screen layout and navigation are clear and easy to use.
* Interface are simple and clear, user can easy to manage habit and group.
* Icons that indicate the actions should be easy to understand and users will not meet any troubles to recognize the feature of application.
  1. Reliability
* The number of sending notification failure is 1 time per 200 notifications.
* The data should be backed up every day.
  1. Availability
* User connects to internet to login into the application.
* User uses offline after login and connect internet to push data to the server.
  1. Security
* Private: Each role of user has a specific permission to interact with the system.
* Only admin can manage user and feedback of user.
  1. Maintainability
* The code is easy to maintain and upgrade.
* Maintain the whole system every 1 months.
* Accidental problem will be resolved within acceptable time.
  1. Portability
* The website is running on Windows 7 or above.
* The mobile application running on a device with API 23 or above.
  1. Performance
* All requests are handled in maximum time of 5 seconds.

1. Conceptual Diagram

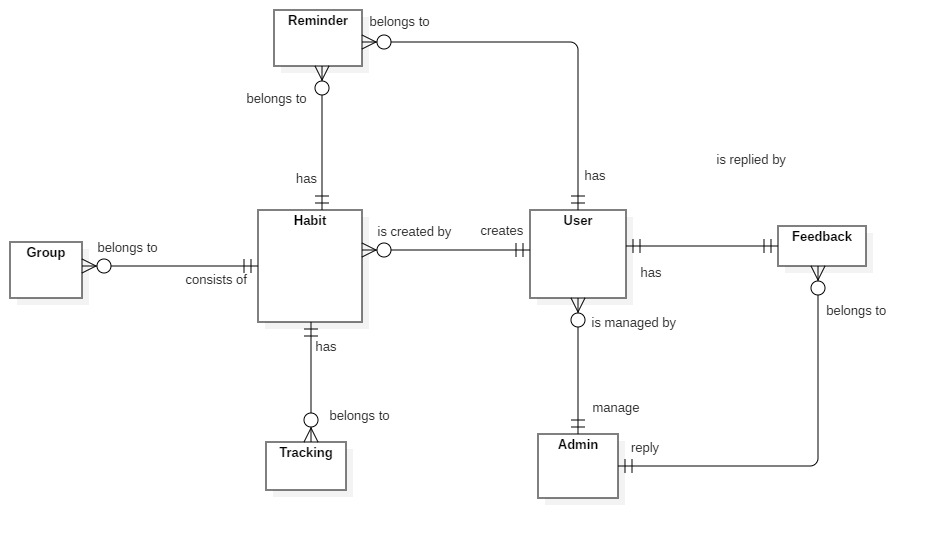


Figure 3 - Conceptual Diagram

***Data Dictionary***

|  |  |
| --- | --- |
| Entity Data dictionary: describe all content of all entities | |
| Entity Name | **Description** |
| App User | Abstract entity describes a user in system |
| User | Contain the user information. |
| Admin | Contain the admin information. |
| Feedback | Contain the feedback information. |
| Habit | Contain the habit information. |
| Notification | Contain the notification information. |
| Tracking | Contain the vehicle information. |
| Reminder | Contain the reminder information. |
| Group | Contain the group information |
|  |  |
|  |  |

Table 12 - Conceptual Diagram Dictionary

1. **Software Design Description**
2. Design Overview

* This document describes the technical and user interface design of MSSC System. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.
* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity. - Document overview:
  + Section 2: gives an overall description of the system architecture design.
  + Section 3: gives component diagrams that describe the connection and integration of the system.
  + Section 4: gives the detail design description which includes class diagram, class explanation, and sequence diagram to details the application functions.
  + Section 5: describe screens design.
  + Section 6: describe a fully attributed ERD.
  + Section 7: describe algorithms.

1. Web architecture design
   1. Web Application architecture description

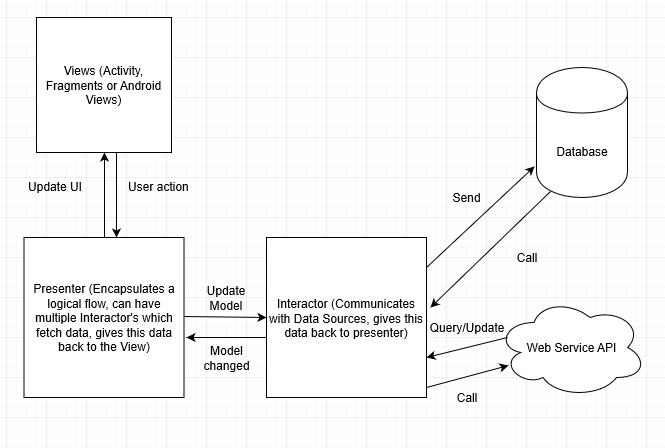


Figure 4 - Web Application Architecture Description

In Web Application, the system is developed under J2EE MVC architecture style. We choose this architecture for Web application because of in scope of the 4 - member team, MVC architecture makes it easier to split the big project into small modules and make it easier to assign each module for members in our team.

This project follows MVC architecture with following components:

* **Servlet (Controller)** is the parts of the application that acts like event handler to handle user interaction. Typically, the controller reads data from a request and calls appropriate Business’s method, then selects view to return to user.
* **JSP/HTML (View)** is the parts of the application that handles the display of the data. The selection of View is under control of Controller.
* **Business** is the parts of the application that do business processing to solve domain problems.
* **Model** is the parts of the application that acts like a data transfer object between the system and database.
* **Web Service** is the parts of the application that acts like an event handler for web and mobile communication via REST method.
  1. Mobile Application architecture description



Numerous architectural patterns appeared, but MVP (Model View Presenter) can fulfill the complete requirements of Android developers.

MVP is one of the patterns, which Android community prefers at this time.

In Android, the application should be easily extensible and maintainable. Therefore, in order to maintain the level, it is important to define separated layers well. And, MVP makes things easier for developers and it makes the views independent of the data source.

* **The Model**
* It represents the layer, which holds the business logic as well as controls how data is created, stored and modified. In Android, it is a data access layer, for example, database API or Remote server API.
* The Model consists of components that are responsible for functionalities like for generating, storing, exposing and fetching the data.
* All these functionalities usually perform in the background thread. Because, these functionalities could be time-consuming, and they can potentially block the main thread UI.
* **The View**
* It is a passive interface, which displays data, and the routes user actions to Presenter.
* In Android, the View is represented by Activity, Fragment or View.
* The View component contains a visual part of the application.
* The View contains the UI and it does not contain any logic or knowledge of the displayed data.
* **The Presenter**
* The Presenter is in between Model and the View. And, it triggers the business logic, and lets to know ‘the View’ when to update.
* It recovers data received from the Model and shows it in the View.
* It interacts with the Model, then fetches and transforms the data from the Model to update the view.

1. Component Diagram



Figure 5 - Web Component Diagram

| **Components Dictionary: Describes components** | |
| --- | --- |
| Web Apllicaiton | Component to controll the system and process request from mobile. Contains sub component: Model, Web Service, Controller. |
| View | Component that display data |
| Controller | Component of website to handle request from web |
| Service | Component to handle business logic of Website component |
| Repository | Component that communicate with database |
| Data Model | Component that do the interaction between the system and database. Contain sub components: Repository and Service |
| Server Database | Component that store data of system |

Table 13 - Web Components Dictionary



Figure 6 - Mobile Component Diagram

| **Components Dictionary: Describes components** | |
| --- | --- |
| Android Apllicaiton | Component to controll the system and process request from mobile. Contains sub component: Model, Android Service, Controller. |
| View | Component that display data. |
| Controller | Component of application to handle request from application. |
| Android Service | Component to handle business logic of application component. |
| Model | Component that do the interaction between the system and database. Contain sub components: Repository and Service |

Table 14 - Mobile Components Dictionary

1. DetailedDescription
   1. Class diagram



Figure 7 - Class Diagram

|  |  |  |
| --- | --- | --- |
| CLASS DICTIONARY: DESCRIBE CLASS | | |
| Class Name | **Mapping column with Conceptual diagram** | **Description** |
| App User | N/A | Contains the app user information |
| Admin | Admin | Contains the admin information |
| User | User | Contains the user information |
| Habit | Habit | Contains the habit information |
| Reminder | reminder | Contain the reminder information |
| Feedback | Feedback | Contains the feedback information |
| Tracking | Tracking | Contains the Tracking information |

Table 15 - Class Diagram

* 1. Class Diagram Explanation
     1. *App User*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier of a user |
| username | String | Private | User’s username |
| name | String | Private | User’s name |
| password | String | Private | User’s password |
| fullName | String | Private | User’s full name |
| phone | String | Private | User’s phone |
| email | String | Private | User’s email |
| roleId | Integer | Private | Id of user’s role |
| isActive | Boolean | Private | User’s status |
| **Method** | **Return Type** | **Visibility** | **Description** |
| getter | Attribute type | Public | Get value of attribute |
| setter | Void | Public | Set value for attribute |
| checkLogin | Boolean | public | Check user login the system |

Table 16 - App User

* + 1. *User*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier of a user |
| username | String | Private | User’s username |
| password | String | Private | User’s password |
| fullName | String | Private | User’s full name |
| phone | String | Private | User’s phone |
| email | String | Private | User’s email |
| roleId | Integer | Private | Id of user’s role |
| isActive | Boolean | Private | User’s status |
| **Method** | **Return Type** | **Visibility** | **Description** |
| getter | Attribute type | Public | Get value of attribute |
| setter | Void | Public | Set value for attribute |

Table 17 - User

* + 1. *Admin*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier of a admin |
| username | String | Private | Admin’s username |
| name | String | Private | Admin’s name |
| password | String | Private | Admin’s password |
| fullName | String | Private | Admin’s full name |
| phone | String | Private | Admin’s phone |
| email | String | Private | Admin’s email |
| roleId | Integer | Private | Id of admin’s role |
| isActive | Boolean | Private | Admin’s status |
| **Method** | **Return Type** | **Visibility** | **Description** |
| getter | Attribute type | Public | Get value of attribute |
| setter | Void | Public | Set value for attribute |

Table 18 - Admin

* + 1. *Habit*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier of a habit |
| name | String | Private | Habit’s name |
| createDate | datetime | Private | Habit’s create date |
| endDate | datetime | Private | Habit’s end date |
| startDate | datetime | Private | Habit’s start date |
| finishDate | datetime | Private | Habit’s finish date |
| group | String | Private | Habit’s group |
| description | String | Private | Habit’s description |
| status | String | Private | Habit’s status |
| **Method** | **Return Type** | **Visibility** | **Description** |
| getter | Attribute type | Public | Get value of attribute |
| setter | Void | Public | Set value for attribute |
| getHabitList | List | Public | Get all habit of user |

Table 19 - Habit

* + 1. *Feedback*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier of user |
| content | String | Private | Feedback’s content |
| **Method** | **Return Type** | **Visibility** | **Description** |

Table 20 - Feedback

* + 1. *Reminder*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier of the habit |
| notificationName | String | Private | Notification’s notification name |
| time | Datetime | Private | Notification’s time |
| description | String | Private | Notification’s description |
| **Method** | **Return Type** | **Visibility** | **Description** |
| getter | Attribute type | Public | Get value of attribute |
| setter | Void | Public | Set value for attribute |

Table 21 – Reminder

* + 1. *Tracking*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier of a habit |
| type | Boolean | Private | Tracking’s type |
| thisDay | datetime | Private | Tracking’s this day |
| **Method** | **Return Type** | **Visibility** | **Description** |
| getter | Attribute type | Public | Get value of attribute |
| setter | Void | Public | Set value for attribute |

Table 22 - Tracking

* 1. Interaction Diagram
     1. *Add new habit*

*Summary: this diagram show process of user adds new habit*



Figure 8 - Sequence Diagram for add new habit

* + 1. *Edit habit*

*Summary: this diagram show process of user edits a habit.*



Figure 9 - Sequence Diagram for edit habit

* + 1. *Delete habit*

*Summary: this diagram show process of user deletes a habit*



Figure 10 - Sequence Diagram for delete habit

* 1. Activity diagram



Figure 11 - Activity Diagram for habit

## Database Design



### Entity Relationship Diagram

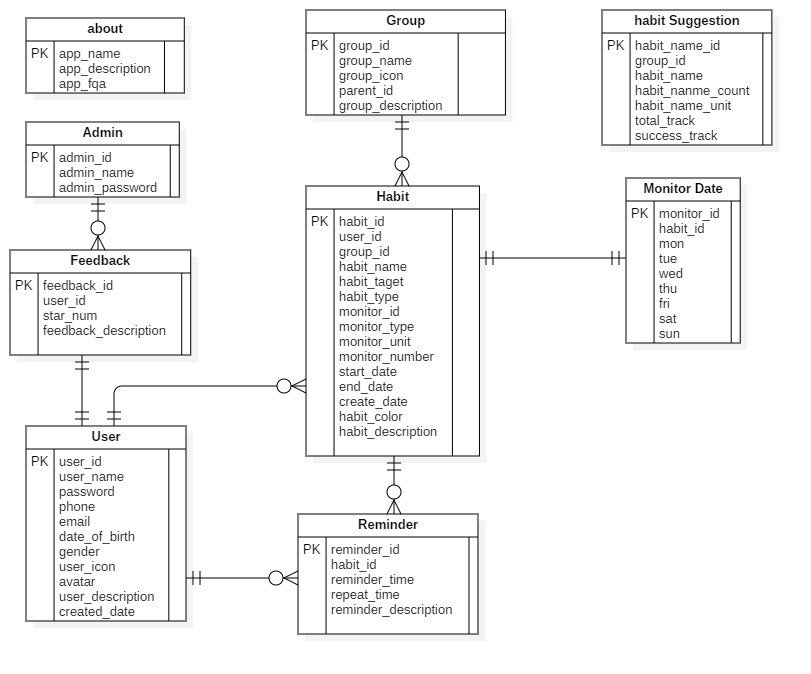


Figure 12 – Entity Relationship Diagram

### Data dictionary

|  |  |
| --- | --- |
| **DATA DICTIONARY: DESCRIBE CONTENT OF ALL TABLES** | |
| **Table name** | **Description** |
| Account | Contains the account information |
| Work Order | Contains the work order information |
| WorkOrderRecord | Contains the information of a user that modified in work order |
| WorkOrderStatus | Contains the work order’s status information |
| WorkOrderCategory | Contains the work order’s category information |
| Priority | Contains the work order’s priority information |
| Equipment | Contains the equipment information |

Table 23 - Entity Relationship Diagram Dictionary

## Database Relationship Diagram



### Physical diagram

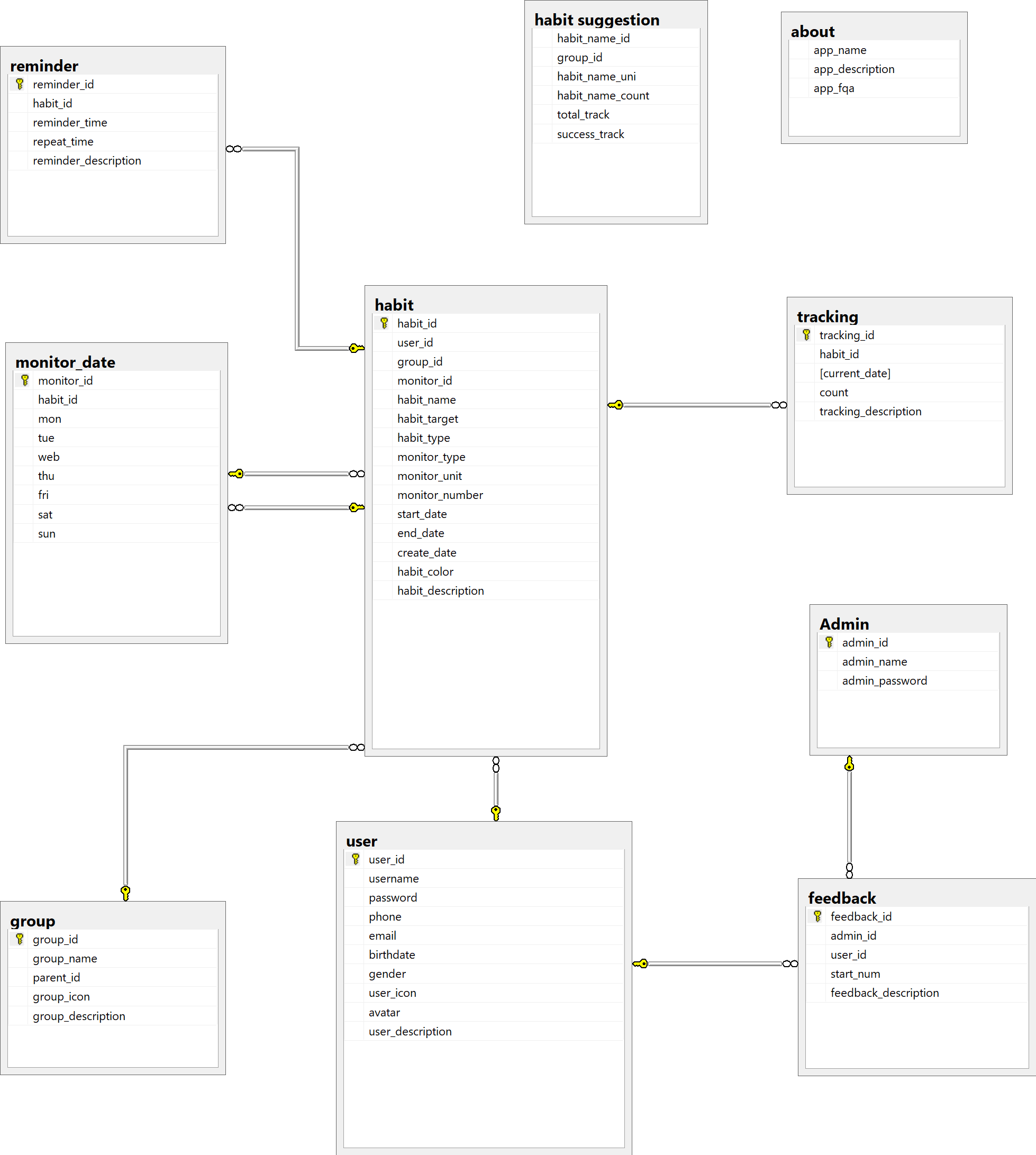


Figure 13 - Physical Diagram

### Data dictionary

|  |  |
| --- | --- |
| **DATA DICTIONARY: DESCRIBE CONTENT OF ALL TABLES** | |
| **Table name** | **Description** |
| User | Contains the user information |
| Admin | Contains the admin information |
| Habit | Contains the habit information |
| Feedback | Contains the feedback information |
| Group | Contains the group information |
| Monitor date | Contains the monitor date information |
| Tracking | Contains the tracking information |
| Reminder | Contains the reminder information |
| Habit suggestion | Contains the habit suggestion information |
| About | Contains the about information |

Table 24 - Physical Diagram Dictionary

## Algorithms

* 1. Recommend the best habits for users by linear programming algorithm
     1. Definition

This algorithm is intended to suggest habits that are commonly used by the community and that are comparable to the current user.

* + 1. Define problem

Helps newcomers or those who have long participated in good habits or give up bad habits to improve themselves.

* + 1. Solution

Supposed

C1 is the total tracking of times the user's habit <30 times.

C2 is the total tracking of the habit from 30 to 60 times of the user.

C3 is the total tracking of the habit from> 60 times of the user.

Based on a habit:

We have:

a: is the number of successful tracking of the habit

b: is the total tracking of the habit (success or failure)

Supposed

p: The habit is done successfully

p = a / b with a / b >= 0.8

f: The habit is done unsuccessfully

f = a / b with a / b < 0.8

Supposed

h: is the difficulty of the habit

k1: The habit is done successfully

k2: The habit is done unsuccessfully

We have:

Difficulty of the habit =

⬄

If

h >= 0.8 -> easy habit

0.5 < h < 0.8 -> medium habit

h < 0.5 -> difficult habit

Then we will recommend for user as below:

For c1, the easy habit is suggested

For c2, the medium habit is suggested

For c3, the difficult habit is suggested

Supposed

d1, d2, d3 are suggestions: easy, medium, difficult

x (n) is a widely used habit, ∀n ∈ N \* (x1 most users, x2 many second ...)

We have the following formula:

If c1 will be applied:

If c2 will be applied:

If c3 will be applied:

Supposed:  
t: is the same display characters are sorted in decreasing turn by user.   
Based on user criteria, we have the following general formula:

* + 1. Complexity

In total, the complexity of this algorithm is **O ().**

* + 1. Example

For example, 10 users use a habit "gym"

User1: Habit "Gym" has a daily type

Start date: 10/10/2018

Current day: 20/10/2018

* Day of implementation is 10 days, of which 8 days is successfully completed, 2 days is failure.

a / b = 0.8 => successful implementation of the habit (Pass)

User2: Habit “Gym” has a yes/no type

Start date: 1/10/2018

Current day: 15/10/2018

The implementation date is 15 days, of which 9 days are yes, 6 days are no.

a / b = 0.6 => implementation of the failure habit (Fail)

Supposed:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | User 1 | User 2 | User 3 | User 4 | User 5 | User 6 | User 7 | User 8 | User 9 | User 10 |
|  | Pass | Fail | Pass | Pass | Fail | Pass | Pass | Pass | Pass | Pass |
| Habit | gym | gym | gym | gym | gym | gym | gym | gym | gym | gym |

8: Pass

2: Fail

* h >= 0.8 -> easy habit

Example user A has done 2 habits with a tracking number is 15 => applies d1

Assume the following top habit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Habit 1 (top1) | Habit 2 (top2) | Habit 3 (top3) | Habit 4(top4) | Habit5(top5) |
| Uses | 553 | 500 | 465 | 326 | 254 |
| Difficulty of the habit | Difficult | Medium | Difficult | Easy | Difficult |

User A will be offered Habit 4

* + 1. Flowchart

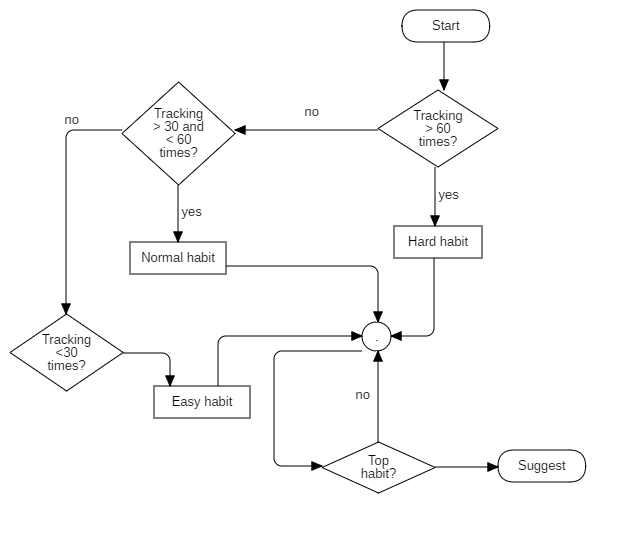


Figure 14 - Linear Programming Algorithm