HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Requirement Specification

Version 1.0

DE-LIPO

Subject: ITSS SOFTWARE DEVELOPMENT

*Authors:*

Pham Le Duc

NGUYEN THI HONG ANH

*Hanoi, December, 2019*

Table of contents

Table of contents 1

1 Introduction 2

1.1 Objective 2

1.2 Scope 2

1.3 Glossary 3

1.4 References 6

2 Overall Description 7

2.1 Actors 7

2.2 Overall use case diagram 8,9

2.3 Business processes 9

3 Specific requirements 9

3.1 Specification of Use case UC001 - “Check in by card" 9

3.2 Specification of Use case UC002 - “Check in by ticket" 10

3.3 Specification of Use case UC003 - “Check card validation" 11

3.4 Specification of Use case UC004 - “Adjust balance" 11

3.5 Specification of Use case UC005 - “Check out by ticket” 11

3.5 Specification of Use case UC006 - “Check out by card” 12

3.5 Specification of Use case UC007 - “Check validation of ticket” 13

3.5 Specification of Use case UC008 - “Control opening and closing gate” 14

4 Supplementary specification 15

4.1 Functionality 15

4.2 Usability 15

4.3 Other requirements 16

# Introduction

## Objective

This document presents the detailed descriptions for the Automated Fare Collection system, user group and their usable function at run time. This document also describes the objectives and features of the system, interfaces, and constraints of the system in response to external action.

## Scope

The software’s goal includes managing the passengers in checking in and checking out of the station. Passengers have two ways to check in and checkout by using tickets or prepaid cards. A card scanner is used to scan card ID, while a ticket recognizer is used for tickets.

Whenever passengers enter or exit the station, if the gate is opened, the system displays “Opening ticket/card”, otherwise “Invalid ticket/card” with a reason. In any case, the program always displays with some basic information of the ticket/card such as id, balance / valid until, status. The station of embarkation is recorded on a prepaid card when a passenger enters the platform area through the automated fare collection system in that station. At this time, if the balance on the card is less than the base fare, the gate is closed to prevent him/her from entering the platform area. When he/she leaves the platform area through the automated fare collection systemin the station of disembarkation, a balance adjustment is processed. Namely, the fare is calculated and is subtracted from the balance on the card. At this time, if this balance is less than the amount of the fare, the gate is closed to prevent him/her from leaving the platform area. (Trang, 2019)

The passengers can only interact with the system through putting tickets / cards. Meanwhile, the system handles validation as well as sending data to the gate, while displaying corresponding information to the passengers. If there is an error, passengers will be notified immediately. Ticket recognizer and Card Scanner are served as actors which help the system in validating the tickets / cards, so they can be replaced easily. Lastly, the system can block or allow any passengers when embarking / disembarking the station. Therefore, it is crucial that the passengers follow strictly the steps and rules to have a complete and happy trip!

## Glossary

**A.**

*Actor:* Users or other systems / programs interacting and exchanging information with the main system / program externally.

**B.**

**C.**

*Card Scanner:* A machine that takes prepaid cards as input and check card’s ID, Information and send the results back to the system to validate.

*Check in:* A process of station embarkation with some steps such as ticket / card validation, record station embarkation, …

*Check out:* A process of station disembarkation with some steps such as ticket / card validation, calculate and adjust balance,…

**D.**

*Data*: facts and statistics collected together for reference or analysis that can be stored and used by a computer

*Disembarkation***:** The action of leaving a ship / train / aircraft after a journey.

**E.**

*Embarkation:* The action of boarding and alighting from modes of transport including trains or ships.

**F.**

*Flow of events: A sequence of events or actions within an use case.*

*Fare***:** Money that passengers must spend in order to use the subway station. Passengers can pay fare in the Ticket Vending Machine.

**G.**

*Gate:* A system that controls opening and closing the physical gate of the AFC system.

*Graphic User Interface:* Interactive components such as buttons, bars, etc… that users can use to interact with programs or systems.

**H.  
I.**

*Information*: is a resolution of uncertainty, which is abstract. Therefore, in order to transmit information, it needs to be converted into data.

*Input***:** Data that is sent into a system or a program to be processed and executed.

*ID:* A combination of unique numbers used for identification in a system or program.

**J.**

*Java Runtime Environment (JRE):* A set of software tools, which are included for development Java applications. Every Java developer must have JRE in order to compile and run a Java program.

**K.**

**L.**

**M.**

**N.**

**O.**

*One-way ticket:* Tickets that can be used once when checking in. It means that after checking in the first time, it cannot be used to check in anymore, only for checking out once.

**P.**

*Prepaid card:* Card bought at the Ticket Vending Machine, which has a certain amount of money inside to enter the station multiple times using such money. Money can be charged more at the vending machine.

**Q.**

**R.**

**S.**

*Software Development Kit (SDK)***:** A set of software tools that allow some applications to be created for certain software packages or other similar development platforms

*System*: a hardware or software system, or combination, which has components as its structure and observable share data as its behavior

**T.**

*Ticket Recognizer:* A machine that takes tickets as inputs and checks ticket’s information including ID, and send result to the system to validate.

*Ticket Vending Machine:* A machine that helps passengers buy tickets / prepaid cards in order to use the subway station.

*Twenty-fourth hour ticket*: Tickets that can be used in 24h multiple times to check in and check out. After 24 hours, it becomes invalid to check out.

**U.**

*Use case*: A sequence of actions a system performs that yields and observable result of value to a particular actor.

**V.**

**W.**

**X.**

**Y.**

**Z**.

## References

Trang, N. T. (2019, September). *Problem Statement: Subway station mini project.* Retrieved from https://www.dropbox.com/sh/8wvobdwp8sqt1xr/AAA7d5ZBJSu07L3KgMzgGIhXa/Mini-project?dl=0&preview=Mini-project+Topic+v1.3.pdf&subfolder\_nav\_tracking=1

# Overall Description

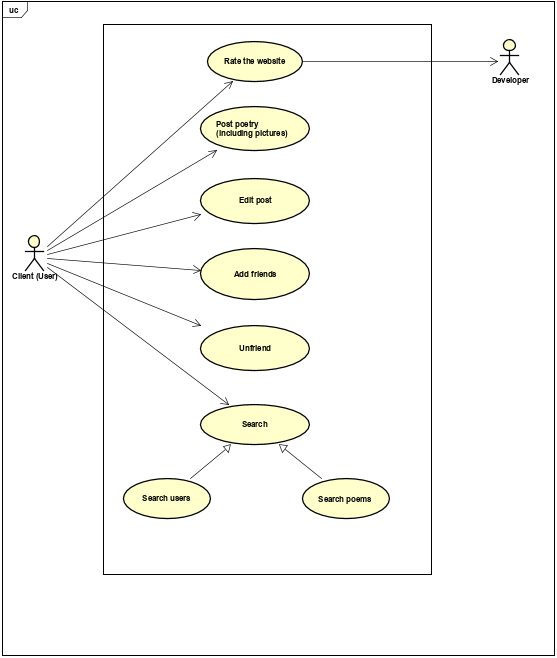
## Actors

There is one main actor in the system: User.

Overall use case diagram

There are two main use cases, which are check in and check out, as every passenger must follow them. When a passenger check-in, they can either check in using tickets or prepaid cards that they have bought from the Ticket Vending Machine. Next, a validation process must be made for the ticket / card in order to confirm that they are valid, and that passenger is qualified to enter the station.

. Check out process is similar, as after travelling, passengers must check out so as to leave the station. They need to put their ticket / card into the system, then the system will validate his information. Specifically, for one-way ticket, if the fare spent is larger than the initial fare then the gate is closed to prevent him from exiting the station. The same process can be applied for the prepaid card, where a balance adjustment process is executed. On the other hand, 24h ticket requires the passenger checking out earlier than the embarkation time in order to check out.



*Main POEM use case diagram*

The gate’s main job is to control whether it should open the gate or not. After validating the ticket / card, the system will send data to the gate for further processing while displaying basic information on screen. If the ticket / card is invalid for some reasons, the gate is closed, otherwise it is opened

## Business processes

# Specific requirements

Details of the use cases given in following sections are specified below.

## Specification of Use case UC001 – “Rate the website”

|  |  |  |  |
| --- | --- | --- | --- |
| Use case code | UC001 | Use case name | Rate the website |
| Actor | User | | |
| Pre-condition | User has already logged into the system and chosen Poem functionality | | |
| Main flow of events (Success) | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1. | User | Enter comment header, comment body and choose rating point | | 2. | System | Check rating point for null | | 3. | System | Set user rating history | | 4. | System | Display user rating post | | 5. | Developer | Get access to rating history database? | | | |
| Alternative flow of events | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | | 2a. | System | Display notice user must choose rating point | | | |
| Post-condition | None | | |

## Specification of Use case UC002 - “Edit post”

|  |  |  |  |
| --- | --- | --- | --- |
| Use case code | UC002 | Use case name | Edit post |
| Actor | User | | |
| Pre-condition | User is logged in and is using poem functionality | | |
| Main flow of events (Success) | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1. | User | Choose post, edit post | | 2. | System | Set user post history | | 3. | System | Display notice: post edited successfully | | | |
| Alternative flow of events | None | | |
| Post-condition | None | | |

## Specification of Use case UC003 - “Post poetry with picture”

|  |  |  |  |
| --- | --- | --- | --- |
| Use case code | UC003 | Use case name | Post poetry with picture |
| Actor | User | | |
| Pre-condition | User is logged in and is using poem functionality | | |
| Main flow of events (Success) | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1. | User | Write text | | 2. | User | Choose picture from library | | 3. | System | Access to user’s library | | 4. | User | Press save post | | 5. | System | Set user activity history | | 6. | System | Display post on user’s feed | | | |
| Alternative flow of events | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | | 3a. | System | System is not granted access to user’s library | | | |
| Post-condition | None | | |

## Specification of Use case UC004 - “Add friend”

|  |  |  |  |
| --- | --- | --- | --- |
| Use case code | UC004 | Use case name | Add friend |
| Actor | Actor | | |
| Pre-condition | User is logged in and is using poem functionality | | |
| Main flow of events (Success) | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1. | User | Send friend request | | 2. | System | Display friend request sent | | 3. | System | Set user activity history (sent friend request) | | 4. | User | Receive friend request | | 5. | User | Confirm friend request | | 6. | System | Display notice users are friends | | 7. | System | Set friend list | | 8. | System | Set user activity history | | | |
| Alternative flow of events | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | | 5a. | User | Decline | | 6a | System | Delete status friend request sent | | 6a. | System | Delete user activity history (sent friend request) | | | |
| Post-condition | None | | |

## Specification of Use case UC005 - “Unfriend”

|  |  |  |  |
| --- | --- | --- | --- |
| Use case code | UC005 | Use case name | Unfriend |
| Actor | User | | |
| Pre-condition | User is logged in and is using poem functionality | | |
| Main flow of events (Success) | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1. | User | Confirm unfriend | | 2. | System | Update user’s friend list | | 3. | System | Display add friend option | | 4. | System | Set user activity history | | | |
| Alternative flow of events | None | | |
| Post-condition | None | | |

## Specification of Use case UC006 - “Search user”

|  |  |  |  |
| --- | --- | --- | --- |
| Use case code | UC006 | Use case name | Search user |
| Actor | User | | |
| Pre-condition | User is logged in and is using poem functionality | | |
| Main flow of events (Success) | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1. | User | Type username | | 2. | System | Search database for username | | 3. | System | Display possible users with same/similar username | | 4. | User | Choose user | | 5. | System | Redirect user to searched user’s profile | | 6. | System | Set user’s search history | | | |
| Alternative flow of events | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | | 3a. | System | If no user found, display no search result | | 5b. | System | Can not open searched user’s profile | | | |
| Post-condition | None | | |

## Specification of Use case UC007 - “Search poem”

|  |  |  |  |
| --- | --- | --- | --- |
| Use case code | UC007 | Use case name | Search poem |
| Actor | User | | |
| Pre-condition | User is logged in and is using poem functionality | | |
| Main flow of events (Success) | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1. | User | Insert poem heading | | 2. | System | Search database for matching results | | 3. | System | Display possible results | | 4. | User | Choose post with poem wanted | | 5. | System | Redirect user to post | | 6. | System | Set user’s search history | | | |
| Alternative flow of events | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | | 3a. | User | If no result found, display notice | | 5a. | System | If can not access post, display notice | | | |
| Post-condition | None | | |

# Supplementary specification

## Functionality

* All the traveling certificates’ information must be able for system to read and update data from the provided data file of traveling certificates.
* All system errors ought to be written to a system error log file. Each system error message ought to include an error ID, an error description, a timestamp, and in which component it occurs, respectively. In addition, the user should be notified whether his/her action results in any system error.
* All activity history (i.e. check-in and check-out history) ought to be written to a history file. Each action ought to include an action’s type (i.e. check-in or check-out action), a timestamp, a traveling certificate ID, successfulness of action, a location where the action is taken, and an error ID if need be.
* The display format is of following:
  + Right justification number
  + Left justification text
  + Font: Arial 14, black
  + White background

## Usability

This section lists all requirements that relate to or affect to the system’s usability.

### Environment Compliance

The Java Runtime Environment (JRE) must be installed to execute the program. The Java JRE 8 is the recommended version.

A ticket recognizer, card scanner, and a gate provided with SDKs for communication must connect to system.

### Ease-of-use UI/UX

A powerful easy command-line GUI is provided to test the program.

### Help

Each feature of the system should give detailed instructions to the users for locating, specifying, and fixing the occurred errors.

## Other requirements

### Reliability

#### Availability

The system ought to be available 24 hours a day, 7 days a week. There should be no more than 5% down time.

#### Mean Time Between Failures

Mean Time Between Failures ought to exceed 700 hours*.*

### Performance

#### Single-User

The system shall support up to one (1) user at a time.

#### Data Access Response Time

The system should access to data file with no more than a thirty-milli-second (30ms) latency.

#### Transaction Response Time

The system must complete 75% of all transactions within one (1) second.

### **Supportability**

### Update to the system should be done by manually