Đỗ Hùng Cường

ITITIU13170

Thesis outline

0/ Abstraction: summary of all thesis – viết sau

List of table

List of figure

Definition

|  |  |
| --- | --- |
| Term | Descriptions |
| Guest |  |
| Customer |  |
| Administrator |  |
| HTML |  |
| CSS |  |
| Boostrap |  |
| Jquery |  |
| IDE |  |
| UML |  |
| Linux |  |

1/Table of content

I/ Introduction

1/ Situation without management system

* Customer must come to hotel to book room => if huge number of customer come at the same time => have to wait => waste time and money => uncomfortable
* Booking based on pen and paper => not convenient for both the customers and the receptionists
* Management is very difficult *(specially in enormous hotels)*
* Reservation might cause a lot of risk *(invalid information of customers, wrong information of rooms or bookings)*

2/ Difficulties with old management system

* Performance might be very bad
* Could overload or run extremely slow when a huge number of users access at the same time
* Look and feed *(User interface was not designed beautifully)*
* Not pleased to use

3/ Modern HB System

* Many deluxe hotels or five-star hotels in the world *(****Marriott International, Hilton Worldwide or InterContinental Hotels Group)*** already have their own hotel booking systems.
* Friendly user interface
* High performance
* Ability to track the behavior of customers.
* The administrators, the managers or hotel owners could know what customer had done on their websites. *(which pages customers clicked on, how long customers stayed at each page, which rooms, which services that customers had searched, booked, ordered or send the feedbacks)*
* Based on the data collection, the systems will automatically suggest what customers may like, recommend which rooms customers should book.
* The hotel owners can improve their hotel business based on the information collected by their systems.

4/ New Technology

Chỉ giới thiệu lí thuyết – tại sao nên dung

a/ Mongodb:

- NoSQL, open-source database

- stores data in JSON-like documents that can vary in structure

- Related information is stored together for fast query

- Dynamic schemas => can create records without first defining the structure

- can change the structure of records simply by adding new fields or deleting existing ones.

- represent hierarchical relationships, to store arrays, and other more complex structures easily

- Documents in a collection need not have an identical set of fields and denormalization of data is common.

- MongoDB was also designed with high availability and scalability in mind, and includes out-of-the-box replication and auto-sharding.

b/ Nodejs

* Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.
* Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.
* Node.js package ecosystem, npm, is the largest ecosystem of open source libraries in the world

c/ Express:

* Web Applications

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

* APIs

With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

* Performance

Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

d/ Angular 2:

5/ Goals and Scope

* Thu nghiem cong nghe MEAN + Spring implements he thong modern HBS
* Includes some inherited features from those five-star hotel booking systems.
* Online single page application with high performance
* Dynamically loading
* Cross-platform system runs well with all operating system.
* Friendly user interfaces
* Supports almost features for hotel bookings & reservations management.
* Ability to track user’s behavior

II/ Background

Lí thuyết công nghệ + mô hình cấu trúc

1/ MEAN

2/Spring

II/ Software Requirement

1/ System Overview

* Hotel Bookings & Reservations System is a web application running on 2 servers
* 2 servers are running at the same time => each server doesn’t have to do a lot of job.
* The main architecture is using MEAN stack technology and J2EE with Spring MVC framework.
* MEAN stack technology => becomes an online single page application with high performance
* Nodejs and express framework => RESTFULL web service + Angular 2 => Dynamically loading + user tracking
* Java => becomes a cross-platform system runs well with all operating system.
* Spring MVC => most powerful java framework => flexible and loosely coupled web applications
* HTML5 + CSS3 + Bootstrap + AngularJS + Angular 2 => Friendly user interfaces

=> comfortable, easy to use.

* 42 primary feature and hundreds of small features.

2/ Feature

* There are many features that my system support for each role *(List functions of each role: guest, customer and admin)*

a/ Guests;

+ view introduction and gallery of the hotel

+ send reservation form, contact with administrators.

+ view, search the rooms or the items in the restaurant which they would like to see more details.

+ register an account to become a customer.

b/ Customer

+ do anything which the guests can do.

+ login to the system to book room or cancel it

+ rate the room, send feedback,

+ check profile

+ view transaction history.

+ With data collection feature, customers were tracked => the system can suggest the recommendation rooms for the customers.

c/ Administrator

+ login to the website and go to their dashboard to manage the hotel

+ check his profile, add, update and delete rooms or other services in the restaurant

+ receive the request of customers and reply them with several available email templates. + manage the users, view information and activity of users or ban them if they did something unacceptably.

+ thank to follow-users feature, administrator is able to see which page customers clicked, how long they stayed in each page, which keyword they used to search, which image they used to click on

+ view the chart with the statistics of visitor from country.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Feature** | **User** | **Description** |
| 1 | Register | Guest, Admin |  |
| 2 | Login | Guest, Admin |  |
| 3 | Logout | Customer, Admin |  |
| 4 | View Rooms | Guest, Customer, Admin |  |
| 5 | View Restaurant | Guest, Customer, Admin |  |
| 6 | Search for Room | Guest, Customer, Admin |  |
| 7 | Search for Food, Drink | Guest, Customer, Admin |  |
| 8 | View gallery of hotel | Guest, Customer |  |
| 9 | View introduction of hotel | Guest, Customer |  |
| 10 | Filer rooms | Guest, Customer, Admin |  |
| 11 | Filer food or drink | Guest, Customer, Admin |  |
| 12 | Send contact | Guest, Customer |  |
| 13 | Send reservation form | Guest, Customer |  |
| 14 | Book room | Customer |  |
| 15 | Cancel room | Customer |  |
| 16 | View profile | Customer, Admin |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Feature** | **User** | **Description** |
| 17 | Edit profile | Customer, Admin |  |
| 18 | Change password | Customer, Admin |  |
| 19 | View activity | Customer |  |
| 20 | Send feedback | Customer |  |
| 21 | Dashboard management | Admin |  |
| 22 | Receive notification | Admin |  |
| 23 | Send message | Admin |  |
| 24 | View users | Admin |  |
| 25 | Manage users | Admin |  |
| 26 | Ban users | Admin |  |
| 27 | Add new room | Admin |  |
| 28 | Delete room | Admin |  |
| 29 | Update room | Admin |  |
| 30 | Add food or drink | Admin |  |
| 31 | Remove food or drink | Admin |  |
| 32 | Update food or drink | Admin |  |
| 33 | Update profile image | Admin |  |
| 34 | Follow users | Admin |  |
| 35 | Send feedback & rate hotel | Customer |  |
| 36 | Send feedback & rate room | Customer |  |
| 37 | View customer activity | Admin |  |
| 38 | View statistic of visit times | Admin |  |
| 39 | View recommendation room | Guest, Customer |  |
| 40 | View related room | Admin |  |
| 41 | View top of rooms | Guest, Customer, Admin |  |
| 42 | Email template | Admin |  |

3/ Use case:



* There are 3 actors using the system: guest, customer and administrator

4/ User story:

Write user story:

* As a guest, I can register a new account so that I can login to the system
* As a guest, I can view the rooms so that I can see the details of the rooms, watch the image of the rooms.
* As a guest, I can view the food or drink in the restaurant of the hotel so that I can see the details, watch the images of each item in the restaurant.
* As a guest, I can view introduction and gallery page so that I can see the information of the hotels and watch the image gallery of the hotel.
* As a guest, I can send contact to the administrator so that I can write what I want to communicate with him and wait for his response.
* As a guest, I can view the recommendation rooms so that I can see which room that the system automatically suggests me book.
* As a customer, I can login to the system or logout so that I can use more features.
* As a customer, I can edit my profile so that I can change my personal information.
* As a customer, I can book room so that when I come to the hotel, this room belongs to me,
* As a customer, I can send a feedback about a room or about the whole hotel services so that I can rate the star of service and comment or complaint my opinion.
* As a customer, I can view my activity so that I can see the transaction history, what I have done, what I interacted with the hotel.
* As an administrator, I can login to the system or logout so that I can use admin features.
* As an administrator, I can edit my profile so that I can change my personal information.
* As an administrator, I can manage the rooms so that I can view the rooms, add a new room, edit a room or delete it.
* As an administrator, I can manage the items in restaurant so that I can view the items, add a new item, edit an item or delete it.
* As an administrator, I can manage users so that I can view user information, view what they interacted with hotel or delete a user from database.
* As an administrator, I can view my messages and notifications which the guests or customers send to me so that I can interact with them and reply their message.
* As an administrator, I can follow user’s behavior so that I can see what they clicked, what they searched, what they did on the website.
* As an administrator I can view the visitor chart from country so that I can easily compare which is the most visited country, which is the less visited country and another.
* As an administrator I can view the page access chart based on all IP address or single IP address so that I can easily compare which is the most visited page, which is the less visited page and another.
* As an administrator, I can receive the message, the booking request, cancel room request and feedback of the customers so that I can view the information that they send to me and reply them by myself or using some available email templates

Kẻ thêm 1 số table use case quan trọng, ko quan trong đem xuống appendix

III/ Methodology

1/ All Technology used:

List technology used:

* Back end: Java web J2EE + Spring MVC framework, Node.js + Express framework
* Front end: HTML5, CSS3, Javascript, Jquery, Boostrap, AngularJS & Angular 2 framework
* Database: MongoDB, RoboMongo
* UML tool: Edraw
* IDE: VSCode, Eclipse, Netbeans
* Front-end design tool: Adobe Dreamweaver CS6
* Server: npm, tomcat, glassfish
* Code review and analysis: Sonar Lint
* Version Control: Git hub
* Project management: Trello

2/ System Architecture

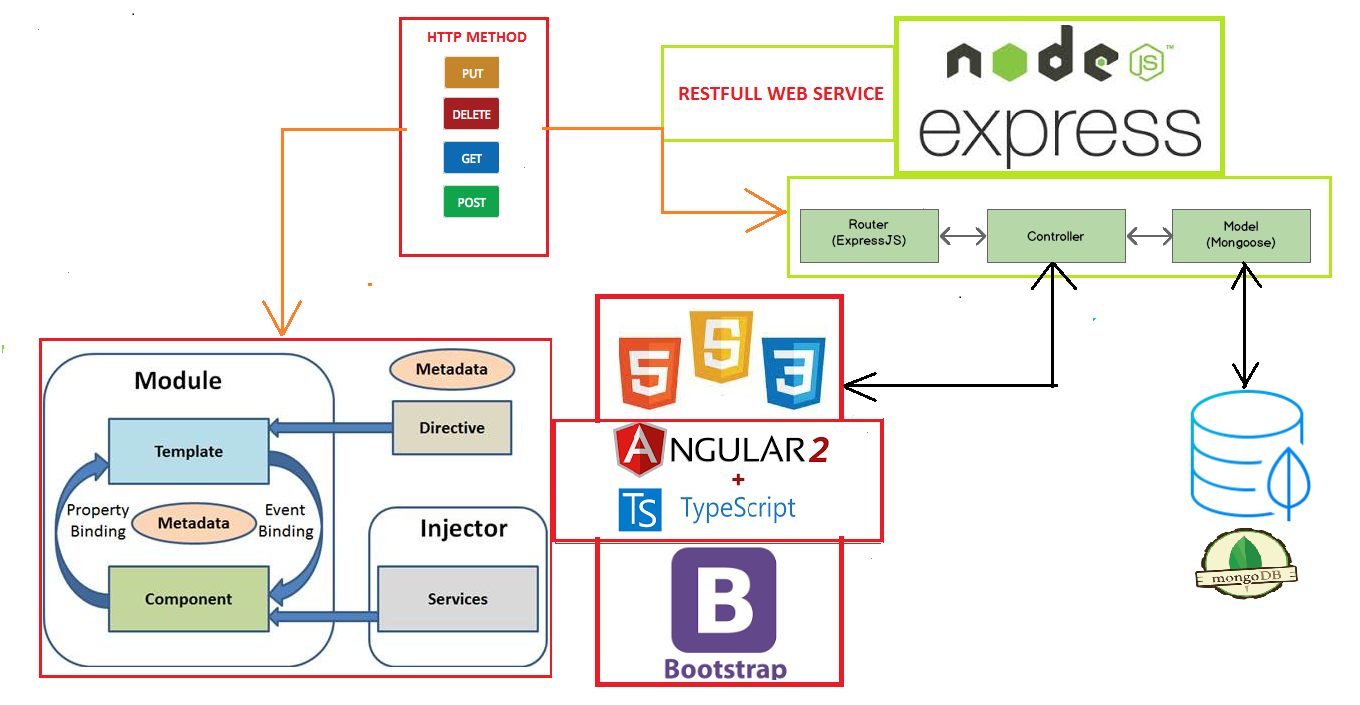
* System includes 2 main architectures:
* MEAN for Customer page
* Spring MVC for Admin Page
* Communicate through API, mongod and file system
* Vẽ mô hình tổng quan: Front end – backend - mongo

a/ MongoDB

What is MongoDB? Why use?

b/ MEAN stack technology:

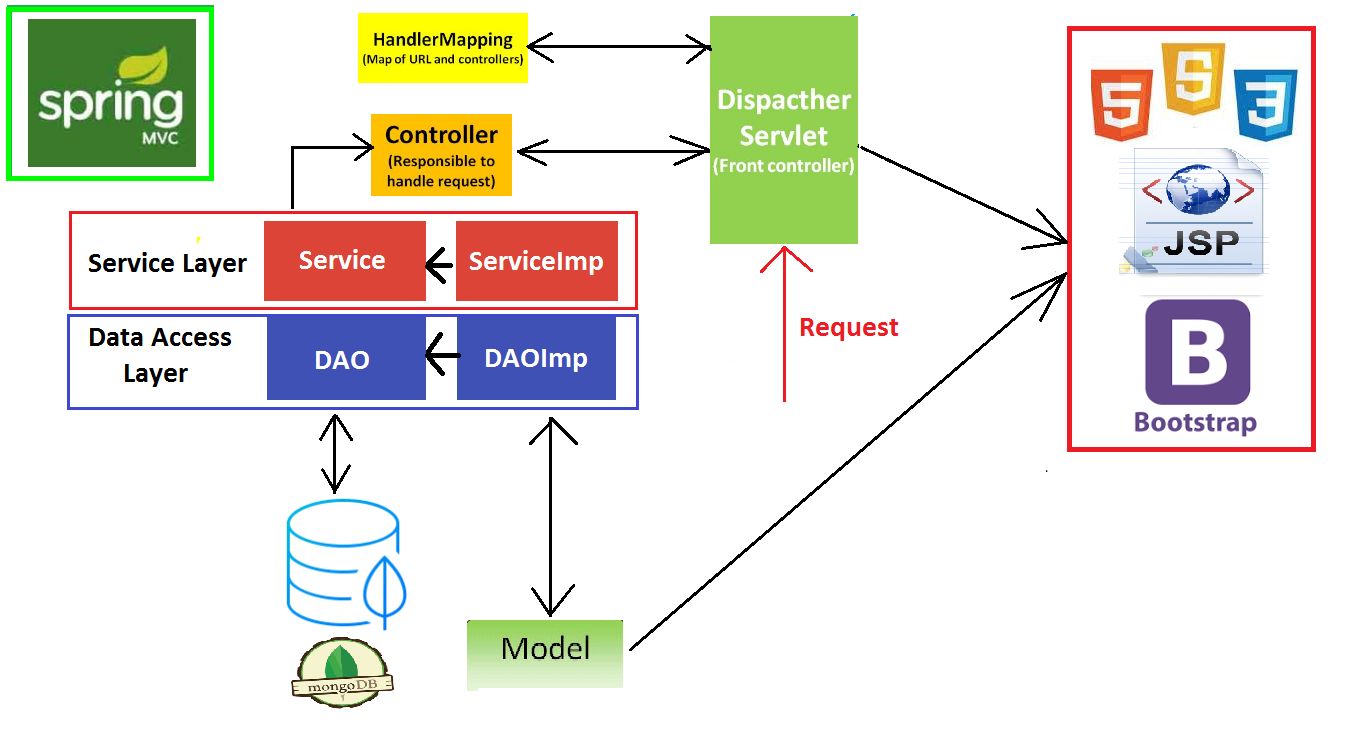
* Mongodb, Angular 2, Express Framework, Nodejs – for Customer & guest page
* Embed Boostrap, HTML, CSS, javascript, Jquery, Angular 2 into HTML file (template)
* The components control template by metadata => provide data binding
* Services provide function for components
* module control all the components, templates, metadata, directives, services
* Mongoose connect database update or retrieve data
* Controller use mongoose to provide functions for router => RESTfull API
* Angular 2 api services interact with RESTfull Webservice
* Request from user => angular 2 routing => determine which page (template+component)
* => display



c/ Spring MVC:

What is Spring MVC? Why use? Describe Spring MVC system?

* Most powerful J2EE framework - use for Admin page
* Embed Boostrap, HTML, CSS, javascript, Jquery, Angular into jsp
* Request -> FrontController -> handle mapping -> read mapping configuration
* FrontController -> Controller -> Service -> ServiceImp -> DAO -> DAOImp -> connect dababase -> update database or retrieve data => Controller use retrieved data -> FrontController -> display
* Controller -> Service -> SericeImpl -> DAO -> DAOImpl -> update to database
* Controller -> Service -> ServiceImp -> API -> APIImp -> interact with API from nodejs server -> api
* MainControler -> RequestMapping -> model map -> display
* RESTController -> api -> angular get api by HTTP methods
* Model: represents the data, provide data type



d/ Diagram

Class diagram – chia rõ class diagram cho frontend và backend

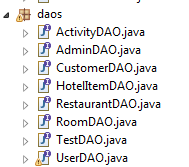
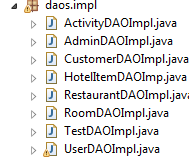
Activity diagram

Database

e/ Details

Spring

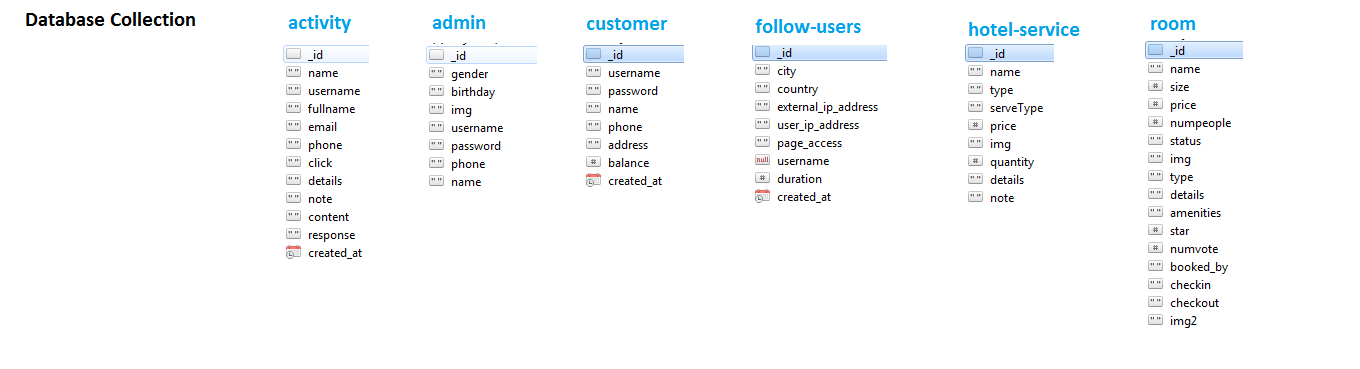
* Maven project – file pom.xml contains dependencies => auto download library needed
* Webapp folder contains resources folder, WEB-INF folder and redirect.jsp file
* resources folder contains all resources for Admin page client side (image, css, js, boostrap, jquery, angular)
* describe how angular work? How embed image, css, js, boostrap, jquery, angular in client side?
* Almost written in java & jsp
* WEB-INF folder is the place storing the view (jspf & jsp files) and configuration files
* describe jspf & jsp. How they work?
* (applicationContext.xml, dispatcher-servlet.xml, web.xml)
* Java Resources include src/main/java, src/main/resources, src/test/java
* Read properties from src/main/resources
* Three Layers Architecture - DAO – Service - Controller
* Using 3 layers architecture => what? Why 3 layers? Separate database layers from controllers + easy to maintain
* MongoDBConnector connect to database mongodb
* Model: represents the data, provide data type
* APIImpl implements API provides methods for ServicesImpl
* API contains method connect with RESTfull Nodejs Server and get the api
* DAOsImpl use MongoDBConnector.java to connect mongodb and implements DAOs provides some methods for ServicesImpl

* ServicesImpl use DAO implements Services provide methods for Controllers
* Controllerss use Services
* Statics package includes AppConst.java and static providers
* AppConst.java contains all Constant variable, array => no hard code
* Statics providers provide some classes contains many static methods (calculate & format date time, send Email, File Upload, edit image, round, StringUtils) for the whole application
* MainController & RESTfullController. How they work?
* How RESTfullController interact with Angular Client side?
* Describe mapping, how Webapp interact with java resources, in MVC architecture
* List all important files and describe => give image
* Ex:
* 

MEAN

* package.json file contains all dependencies => auto download lib
* server.js declares some configuration (router, cookie, app Constanst) and host the server
* app-const.js file contains all constant variables provide for the whole server
* routes determine which page (template+component) is considering and will redirect.
* Nodejs model mongoose connect database => update or retrieve data
* Controller use model, send update request to model or get data from mongoose
* Controller provides function for routers to provide api
* Nodejs + Express framework => RESTfull Web service => provides HTTP methods (GET, PUT POST, DELETE) => api => json
* Embed Boostrap, HTML, CSS, javascript, Jquery, Angular 2 into HTML file (template controlled by component (typescript files)
* The components control template by metadata => provide data binding
* Services are injected to component and provide function for components
* All the components, templates, metadata, directives, services are declared in a module and controlled by it.
* Angular 2 api services interact with RESTfull Webservice
* services receive request from component or return data to component
* Nodejs + Express => RESTfull interact with Angular 2 by HTTP Methods
* List all important files and describe => give image

c/ Mongodb

* 6 collections 

- No entity relationship => don’t need to define the structure

- stores data in JSON-like documents => easily work with javascript, Angular, nodejs

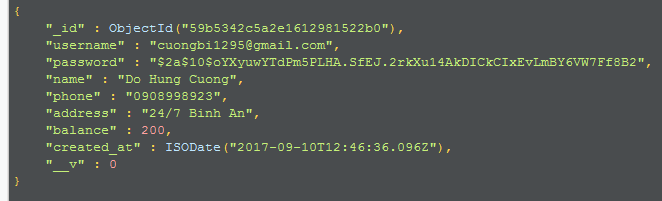
- redundant data but high performance (Related information is stored together for fast query) => Optimize Query Performance

- difficulty in update, when the customer changes his information => have to update all collection that related to this customer or when delete a room …

- However, delete a room or change user information is not usually => reduce update speed to improve query performance is ok

* Show all collections and describe => give image

Ex:



e/ Implementation:

chụp hình 1 vài đoạn code quan trọng

chụp hình chương trình chạy

V/ Experiment and Result

1/ Experiments

* System run well on window
* Test on Linux
* Responsive website => test mobile UI
* Encrypt password
* Recommendation room
* Chart

2/ Evaluation

* Online single page application with high performance => speed? Compare with what?
* Dynamically loading? Why? prove
* Cross-platform system runs well with all operating system. => prove? Run well on window + Linux, + mobile?
* Friendly user interfaces, easy to use => show image
* Supports almost features for hotel bookings & reservations management. => show list features
* Ability to track user’s behavior => show chart, image

Give examples, image to prove

VI/ Conclusion

* Áp dụng MEAN stack, Spring MVC, em thấy như thế nào
* Trong tương lai em sẽ làm gì , them tính năng …, AI …, Machine learning…
* Hotel business is a highly profitable industry but requires huge investment as well as having to meet the customer's demand.
* Management system is really important
* Good management system brings higher profit
* The key is to pleasure the customers
* Friendly user interface
* High performance
* Dynamically loading
* Tracking customer’s behavior
* Improve system day by day
* After thesis => learn a lot of new things
* Learn a lot of technologies: AngularJS, Angular 2, MongoDB, Spring MVC, Nodejs, Express Framework…
* Experience in building single page application
* Working with a lot of frameworks
* Ability to learn new technology
* Improve myself in the future for working in professional environment

VII/ Appendix

1/ Use case bổ sung

Kẻ thêm nhưng bảng use case bổ sung cho phần software requirement, trên phần software requirement nói “để hiểu rõ về … xem phần appendix 1.1, 1.2…

2/ Sequence diagram bổ sung

* Draw some sequence diagram that represent for the whole system

3/ Test Case bổ sung

* Test case table

4 User Manual

* How to use + image

VIII/ Reference