

**MAHARISHI UNIVERSITY**

**OF MANAGEMENT**

|  |
| --- |
| **SWE Project** |

**QUARO**

|  |  |
| --- | --- |
| **Group member** | |
| **Group members** | Dinh Tung Nguyen – 987007 (Mr.)  Aaron Abraha Gezai – 109660 (Mr.)  Habtom Abraha Gebre – 109641 (Mr.)  Samuel Asmelash Tesfaburk – 109696 (Mr.)  Saw Qua Lar – 986226 (Mr.) |
| **Professor** | Obinna Kalu (Prof.) |
| **Project code** | QUARO |

**Fairfield, May 29th, 2019**

**ACKNOWLEDGEMENT**

We are grateful to our professor, Prof. Obinna Kalu because of his enthusiastic guidance. Without his knowledge and instruction, we cannot complete our project.

Fairfield, May 29th, 2019

QUARO Team

# Chapter 1: Introduction

## Project Information

* Project name: **Quaro**
* Project code: **QUARO**
* Product type: **Website Application**
* Start date: **05/27/2019**
* End date: **06/20/2019**

## Introduction

Quaro’s mission is to share and grow the world’s knowledge. A vast amount of the knowledge that would be valuable to many people is currently only available to a few — either locked in people’s heads, or only accessible to select groups. We want to connect the people who have knowledge to the people who need it, to bring together people with different perspectives so they can understand each other better, and to empower everyone to share their knowledge for the benefit of the rest of the world.

## Roles and Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Full Name | Role | Position | Contact |
| 1 | Dinh Tung Nguyen | Developer | Member | dtnguyen@mum.edu |
| 2 | Aaron Abraha Gezai | Developer | Member | agezai@mum.edu |
| 3 | Habtom Abraha Gebre | Developer | Member | hgebre@mum.edu |
| 4 | Samuel Asmelash Tesfaburk | Developer | Member | stesfaburk@mum.edu |

Roles and Responsibilities

# Chapter 2: Software Project Management Plan

## Problem Definition

### Name of this project

* Official name: Quaro.
* Abbreviation: QUARO.

### Problem Abstract

Quaro’s mission is to share and grow the world’s knowledge. A vast amount of the knowledge that would be valuable to many people is currently only available to a few — either locked in people’s heads, or only accessible to select groups. We want to connect the people who have knowledge to the people who need it, to bring together people with different perspectives so they can understand each other better, and to empower everyone to share their knowledge for the benefit of the rest of the world.

### Project Overview

#### The Proposed System

QUARO has a website application.

Web application consists of following features:

* Guest can signup for creating new account and can login with the authorized account for accessing Quaro website.
* Users can view profile.
* Users can post new question.
* Users can answer the question.
* Users can upvote and downvote the question.
* Users can upvote and downvote the another answer.

#### Boundaries of the System

* The system can be used by every people with a computer/laptop with browser (like Chrome, Firefox, Safari…).
* The complete product contains:
* Website for the authorized users.
* All related documents.

#### Development Environment

* **Hardware requirements**

|  |  |  |
| --- | --- | --- |
| Windows | Minimum Requirements | Recommended |
| Internet Connection | Cable, Wi-Fi (4 Mbps) | Cable, Wi-Fi (8 Mbps) |
| Operating System | Windows 7, 8 or 10 | Windows 7, 8 or 10 |
| Computer Processor | Intel® Xeon ® 1.4GHz | Intel® Core i5 2.4GHz |
| Computer Memory | 1GB RAM | 2GB or more |

Hardware Requirements for Server

* **Software requirements**

|  |  |  |
| --- | --- | --- |
| Software | Name / Version | Description |
| Operating System | Window 10 and MacOS | Operating system and platform for development. |
| Environment | Java 8, JDK 8, AngularJS | Specification for developing web. |
| IDE | IntelliJ IDEA Community 2019 | Used for implementing website application and web service. |
| Design Model Tool | StarUML 2.8.1 | Used for creating models and diagrams. |
| Database Management Tool | MySQL Workbench 8.0 | Used for creating and managing database for system. |
| Document & Source  Code Management | Bitbucket | Used for storing and managing document and source code. |

Software Requirements

## Project Organization



### Software Process Model

#### Overall Description



Waterfall model

The Waterfall Model is a sequential (non-iterative) design process used in software development. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of model is used for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model, the testing starts only after the development is complete. In waterfall, model phases do not overlap.

#### Reasons for choosing

This project is developed under Waterfall model to consistent with current situation of our team. We choose this model because of following reasons:

* QUARO is a project which be done in short time.
* Team members have limited skills and experiences since no one has worked with any real project so that all requirements must be well defined.
* All members are used to with Waterfall concept that we can understand all the aspects of project organization.
* Team meetings should be held frequently to keep team coherence.

### Tools and Techniques

* Front-end IDE: IntelliJ IDEA Community 2019.
* Back-end IDE: IntelliJ IDEA Community 2019.
* Front-end technologies: AngularJS combined with HTML, CSS, and Bootstrap… to build website application.
* Back-end technologies: Spring MVC, Hibernate, RESTful Web Service.
* Database: MySQL Workbench 8.0 for database management.
* Source Control: Sourcetree v3.1.2(216) with Bitbucket.org: https://tungnd1707@bitbucket.org/tungnd1707/swe.git

# Chapter 3: Software Requirement Specification

## User Requirement Specification

### User Requirement

* Signup new account.
* Login with the authorized account.
* View profile.
* Post question.
* Answer the question.
* Upvote and downvote question.
* Upvote and downvote answer.

## System Requirement Specification



### External Interface Requirements

#### User Interfaces

* The user interface must be friendly and easy to use.
* The user interface of website displays best on 1366×768 resolution on desktop.

#### Hardware Interfaces

* The system will use the standard hardware and data communications resources of a standard computer.
* Desktop or laptop has ability to connect to the Internet.

#### Software Interfaces

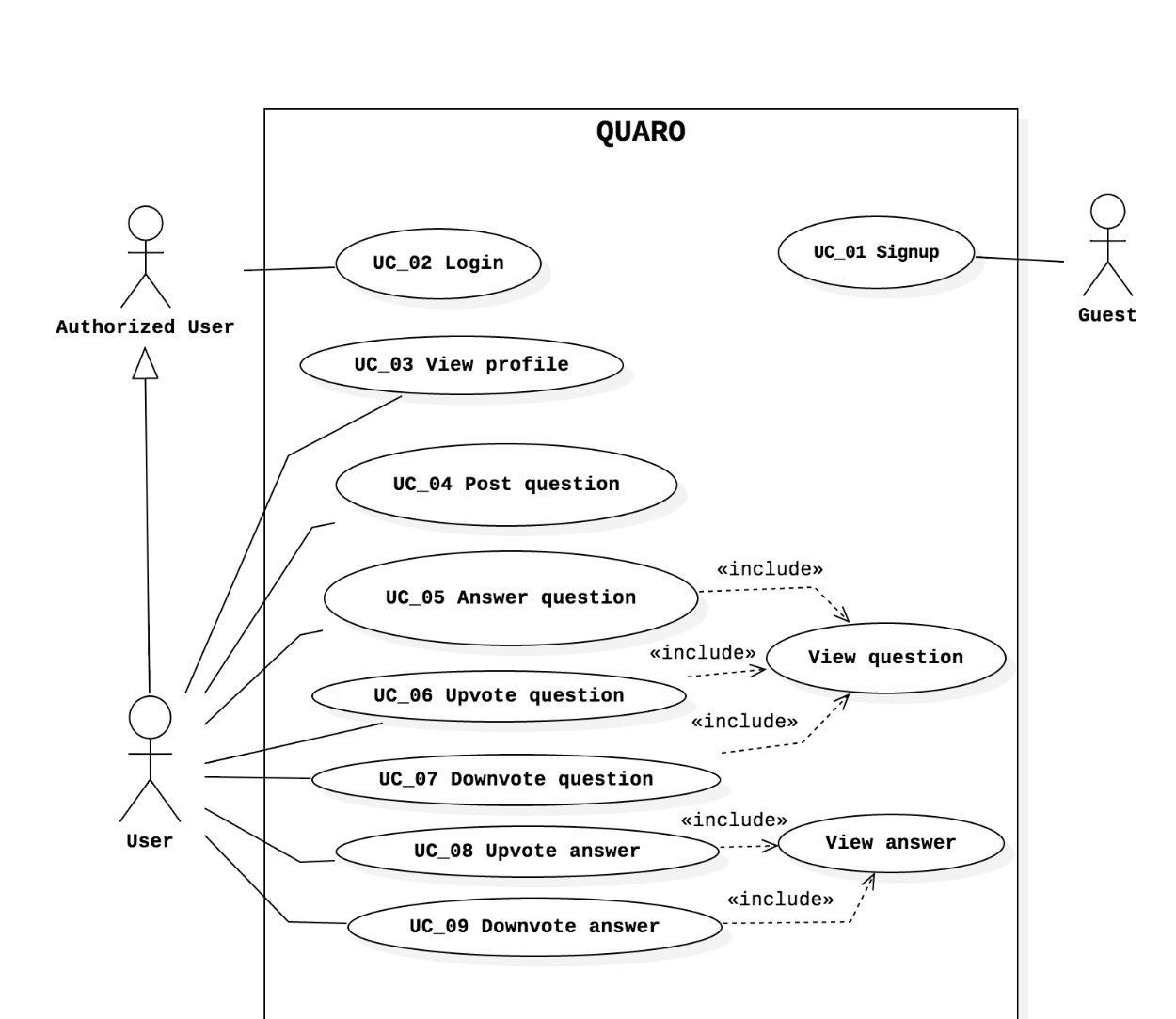
* Website: work with Google Chrome (v74 or above), Safari (v12.1.1 or above) or with any web browser that supports HTML, CSS and JavaScript.

#### Communications Protocol

* Use HTTP protocol 1.1 for communication between the web browser and the web service.
* Use HTTP protocol 1.1 for communication between the mobile application and the web service.

### Functional Requirements

#### System Overview Use Case



System Overview Use Case

#### Use Case Specifications

|  |  |  |  |
| --- | --- | --- | --- |
| No | ID | Name | Actor |
| 1 | UC\_01 | Signup | Guest |
| 2 | UC\_02 | Login | Authorized User |
| 3 | UC\_03 | View profile | User |
| 4 | UC\_04 | Post question | User |
| 5 | UC\_05 | Answer question | User |
| 6 | UC\_06 | Upvote the question | User |
| 7 | UC\_07 | Downvote the question | User |
| 8 | UC\_08 | Upvote the answer | User |
| 9 | UC\_09 | Downvote the answer | User |

List of use case

##### UC\_01\_Signup Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_01 SPECIFICATION** | | | |
| **Use Case No.** | UC\_01 | **Use Case Version** | 1.0 |
| **Use Case Name** | Signup | | |
| **Date** | 05/29/2019 | **Priority** | High |
| **Actor:**   * Guest   **Summary:**   * This use case allows guest to signup the new account.   **Goal:**   * Guest can signup a new account.   **Triggers:**   * Guest clicks on “Sign Up” link.   **Preconditions:**   * Actor has not logged in to the system.   **Post Conditions:**   * **Success**: The new account will be created. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Guest clicks on “Sign Up” link | System shows “Sign Up” form with:   * Email: textbox, email type, required. * Password: textbox, password type, required. * Confirm password: textbox, password type, required. * First name: textbox, required. * Last name: textbox, required. | | **2** | Guest fills information into the form |  | | **3** | Guest clicks on “Sign Up” button  [Alternative 1] | System saves the information and shows success message: “You signup successfully!”  [Exception 1, 2, 3] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | **1** | Guest clicks on “You already had an account? Login now!” link | System shows “Login” form. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | **1** | Required field is missing. | System shows required error message. | | **2** | Invalid email format. | System shows error message: “Email is invalid!”. | | **3** | Email is existed in the system. | System shows error message: “Email is existed!”. |   **Relationships:**   * N/A   **Business Rules:**   * All required fields must be filled. * Email must be unique. * “Email” field must follow format [abc@def.com](mailto:abc@def.com). * After register success, user can login with the account successfully. | | | |

##### UC\_02\_Login Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_02 SPECIFICATION** | | | |
| **Use Case No.** | UC\_02 | **Use Case Version** | 1.0 |
| **Use Case Name** | Login | | |
| **Date** | 05/29/2019 | **Priority** | High |
| **Actor:**   * Authorized user.   **Summary:**   * This use case allows actor to login to the system.   **Goal:**   * Actor can login to the system.   **Triggers:**   * Actor clicks on “Login” button.   **Preconditions:**   * Actor has not logged in to the system.   **Post Conditions:**   * **Success**: Actor has logged in to the system. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Actor clicks on “Login” button. | System shows “Login” form with:   * Email: textbox, email type, required. * Password: textbox, password type, required. | | **2** | Actor inputs email and password. |  | | **3** | Actor clicks on “Login” button.  [Alternative 1] | Actor will login system successfully.  [Exception 1, 2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | **1** | Actor clicks on “Do not have account? Sign up now!” link. | System shows “Sign Up” form. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | **1** | Required fields is missing | System shows required error message. | | **2** | Invalid email format | System shows error message: “Email is invalid!”. |   **Relationships:**   * N/A   **Business Rules:**   * All required fields must be filled. * Email and password must match existing data in the system. | | | |

##### UC\_03\_View Profile Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_03 SPECIFICATION** | | | |
| **Use Case No.** | UC\_03 | **Use Case Version** | 1.0 |
| **Use Case Name** | View profile | | |
| **Date** | 05/29/2019 | **Priority** | Normal |
| **Actor:**   * User.   **Summary:**   * This use case allows user to view profile.   **Goal:**   * User can view profile.   **Triggers:**   * User clicks on “View Profile” link.   **Preconditions:**   * Profile page will be shown.   **Post Conditions:**   * **Success**: User can view profile successfully. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | User clicks on “View Profile” link. | System shows Profile page on screen. |   **Alternative Scenario**:   * N/A   **Exceptions:**   * N/A   **Relationships:**   * N/A   **Business Rules:**   * User can view profile successfully with the correct information. | | | |

##### UC\_04\_Post question Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_04 SPECIFICATION** | | | |
| **Use Case No.** | UC\_04 | **Use Case Version** | 1.0 |
| **Use Case Name** | Post question | | |
| **Date** | 05/29/2019 | **Priority** | High |
| **Actor:**   * User.   **Summary:**   * This use case allows user can post the new question.   **Goal:**   * Actor can post the new question successfully.   **Triggers:**   * Actor clicks on “Post question” button.   **Preconditions:**   * Actor logged in to the system.   **Post Conditions:**   * **Success**: The new question can post successfully. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Actor clicks on “Post question” button. | System shows “Post question” form with:   * Title: textbox, required. * Content: textbox, required. | | **2** | Actor fills out the information. |  | | **3** | Actor clicks on Post button. | System shows message: “Post new question successfully!”.  [Exception 1] |   **Alternative Scenario:**   * N/A   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | **1** | Required fields is missing. | System shows required error message. |   **Relationships:**   * N/A   **Business Rules:**   * N/A | | | |

##### UC\_05\_Answer question Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_05 SPECIFICATION** | | | |
| **Use Case No.** | UC\_05 | **Use Case Version** | 1.0 |
| **Use Case Name** | Answer question | | |
| **Date** | 05/29/2019 | **Priority** | High |
| **Actor:**   * User.   **Summary:**   * This use case allows actor to answer question.   **Goal:**   * Actor can use this method to answer question.   **Triggers:**   * Actor clicks on “Answer” button.   **Preconditions:**   * N/A   **Post Conditions:**   * **Success**: User can answer question successfully. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Actor goes to home page. | System shows the question on screen. | | **2** | Actor clicks on any question. | System shows the question with another answer with:   * Content: textbox, required. | | **3** | Actor inputs the content and clicks on “Answer” button. | System shows the question with the newest answer on screen.  [Exception 1] |   **Alternative Scenario:**   * N/A   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | **1** | Required fields is missing. | System shows required error message. |   **Relationships:**   * N/A   **Business Rules:**   * N/A | | | |

##### UC\_06\_Upvote question Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_06 SPECIFICATION** | | | |
| **Use Case No.** | UC\_06 | **Use Case Version** | 1.0 |
| **Use Case Name** | Upvote question | | |
| **Date** | 05/29/2019 | **Priority** | Normal |
| **Actor:**   * User.   **Summary:**   * This use case can allows actor to upvote question.   **Goal:**   * Actor can upvote question successfully.   **Triggers:**   * Actor clicks on “Upvote” button at the question.   **Preconditions:**   * N/A   **Post Conditions:**   * **Success**: User can upvote question successfully and the number of voting will plus 1. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Actor clicks on “Upvote” button at the question. | System will plus 1 and display the new number of the voting. |   **Alternative Scenario:**   * N/A   **Exceptions:**   * N/A   **Relationships:**   * N/A   **Business Rules:**   * N/A | | | |

##### UC\_07\_Downvote question Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_07 SPECIFICATION** | | | |
| **Use Case No.** | UC\_07 | **Use Case Version** | 1.0 |
| **Use Case Name** | Downvote question | | |
| **Date** | 05/29/2019 | **Priority** | Normal |
| **Actor:**   * User.   **Summary:**   * This use case can allows actor to downvote question.   **Goal:**   * Actor can downvote question successfully.   **Triggers:**   * Actor clicks on “Downvote” button at the question.   **Preconditions:**   * N/A   **Post Conditions:**   * **Success**: User can downvote question successfully and the number of voting minus 1. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Actor clicks on “Downvote” button at the question. | System will minus 1 and display the new number of the voting. |   **Alternative Scenario:**   * N/A   **Exceptions:**   * N/A   **Relationships:**   * N/A   **Business Rules:**   * N/A | | | |

##### UC\_08\_Upvote answer Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_08 SPECIFICATION** | | | |
| **Use Case No.** | UC\_08 | **Use Case Version** | 1.0 |
| **Use Case Name** | Upvote answer | | |
| **Date** | 05/29/2019 | **Priority** | Normal |
| **Actor:**   * User.   **Summary:**   * This use case can allows actor to upvote answer.   **Goal:**   * Actor can upvote answer successfully.   **Triggers:**   * Actor clicks on “Upvote” button at the answer.   **Preconditions:**   * N/A   **Post Conditions:**   * **Success**: User can upvote answer successfully and the number of voting will plus 1. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Actor clicks on “Upvote” button at the answer. | System will plus 1 and display the new number of the voting. |   **Alternative Scenario:**   * N/A   **Exceptions:**   * N/A   **Relationships:**   * N/A   **Business Rules:**   * N/A | | | |

##### UC\_09\_Downvote answer Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_09 SPECIFICATION** | | | |
| **Use Case No.** | UC\_09 | **Use Case Version** | 1.0 |
| **Use Case Name** | Downvote answer | | |
| **Date** | 05/29/2019 | **Priority** | Normal |
| **Actor:**   * User.   **Summary:**   * This use case can allows actor to downvote answer.   **Goal:**   * Actor can downvote answer successfully.   **Triggers:**   * Actor clicks on “Downvote” button at the answer.   **Preconditions:**   * N/A   **Post Conditions:**   * **Success**: User can downvote answer successfully and the number of voting minus 1. * **Fail**: Error message is shown.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Actor clicks on “Downvote” button at the answer. | System will minus 1 and display the new number of the voting. |   **Alternative Scenario:**   * N/A   **Exceptions:**   * N/A   **Relationships:**   * N/A   **Business Rules:**   * N/A | | | |

### Non-Functional Requirements

#### Reliability

* All of data must be accurate.
* Make sure integrity of data.
* Guarantee saving data correct.

#### Availability

* People can use the application every time through internet connection by their computer within internet connection.
* Percent of time available is about 97 – 98%. Because the system uses client/server protocol through internet network, we cannot control cases of force major such as power failure, disconnected network, etc.

#### Portability

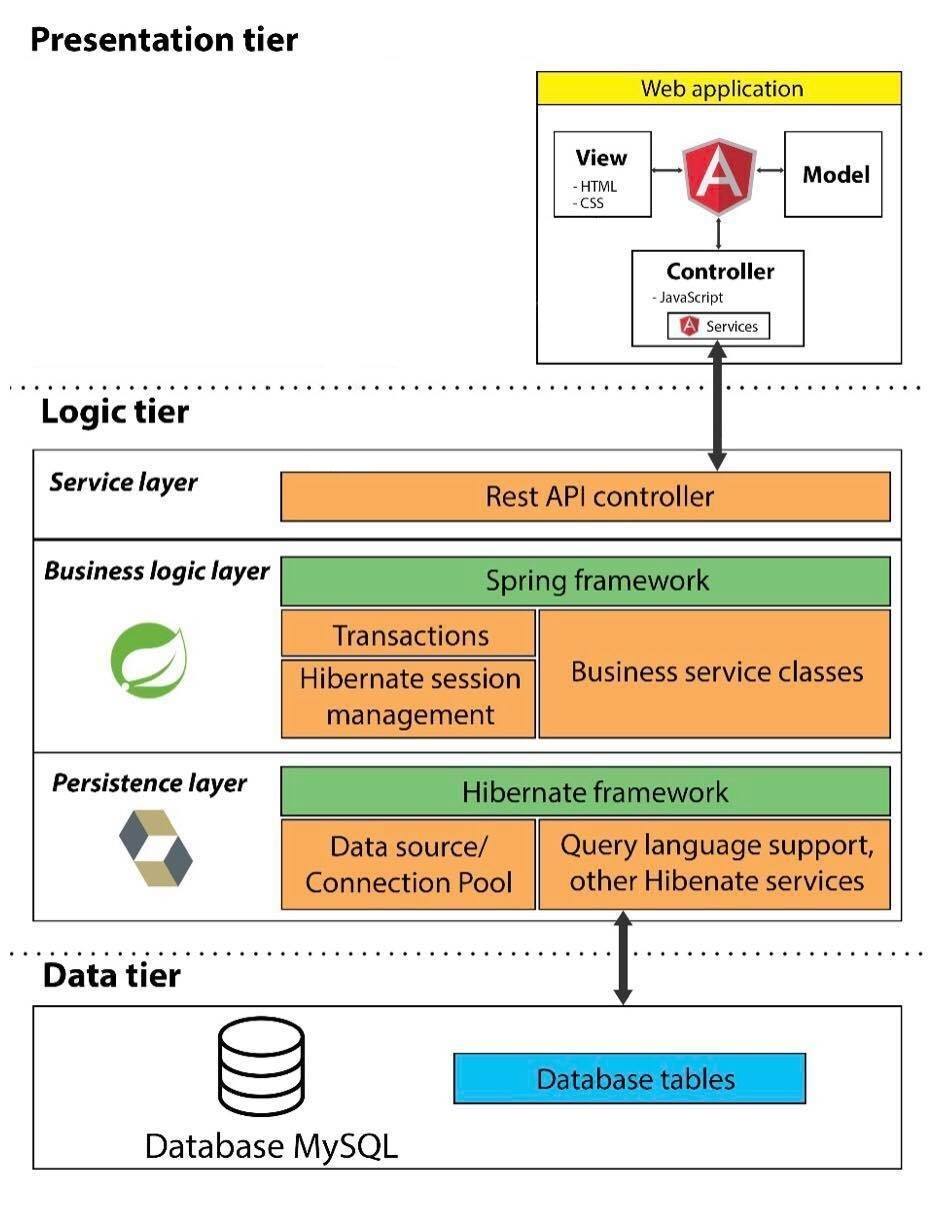
* The software itself is a web application, therefore it can be used in any platform that has a web browser and can connect to the internet.

#### Performance

* Load time :
* Every page should be completely displayed in average 2 seconds and maximum 5 seconds within 4Mbps bandwidth speed and 1 GHz Processing Speed of CPU.
* Compatibility :
* The system should provide elegant response for nowadays common screen resolutions: 1280×800, 1366×768 and 1440×900.

# Chapter 4: System Design Description

## 1. System Architectural Design



System Architectural Design

Three-tier architecture is a client–server software architecture pattern in which the user interface (presentation), functional process logic (business rules), computer data storage and data access are developed and maintained as independent modules. *(source: Wikipedia)*

Our system has 3 essential modules; each module has particular responsibility to operate its function:

* **Presentation tier:**
  + This tier displays information to end-user that they can manage with their data or interact with another user.
  + Web application: based on AngularJS project structure, combine using of HTML, Javascript and CSS to one project. Controller components (Javascript files) interact with service layer of Logic tier to request and receive data from database. Model components use Angular local storage technique to storing client data. View components are a presentation of data in HTML format, triggered by controller’s decision.
* **Logic tier:**
  + Service layer: Use Representational state transfer (REST) web service to provide API for accessing data. RESTful web service is supported by Spring framework by using Spring annotation mapping and configuration.
  + Business logic layer: Use Spring framework to operate system business logic function by Spring technique, includes: transaction management and session management.
  + Persistence layer: Use hibernate framework to manage database accessing operation includes: data source management, connection pool support, query language support…
* **Data tier:**
  + Our system use MySQL to manage database. The persistence layer of Logic tier connects to Data tier by using MySQL connector driver.



### 2. Discussion of chosen AngularJS in presentation tier

AngularJS has many strengths that can meet our system requirement and goal that we have decided to choose this framework. AngularJS strength is about:

#### 2.1 MVC pattern familiar

AngularJS splits web application into MVC components, which is familiar with our team member to understand and manage code.

#### 2.2 Client side advantages

Different with JSP, AngularJS can be developed separately from web services that we can archive:

* More interactivity by immediately responding to users’ actions.
* Execute quickly because they do not require waiting response from server for loading pages.
* Easy to reuse resource and manage HTML template separately from server.

#### 2.3 Modularity

In AngularJS, we can create application combining separated modules and manage each module separately.

For example: Staff or Admin will have different views from User so that we cannot apply one UI for all roles. AngularJS allow us to separates from project to modules so that we can handle UI well although both modules use the same resources.

#### 2.4 Dependency Injection

Different from some other client side framework, AngularJS allow us to design MVC pattern that can inject wrapper, setter, getter or class declaration across the application. AngularJS do it themselves so that we can save the most of development cost.

## Class diagram

