**Project Name: Volunteering For Change**

**Project Group: Trojan**

**Course: Web Application Architecture**

**Submitted To: Joe Brune**

**Project Group Members:**

Ashish Nembang

Sanjeev Bhusal

Rishikesh Mandal

Efrem Tewelde

**Table of Contents**

1. Introduction.

1.1 Purpose.

2. Requirements - Use-Case – Usage Scenarios.

3. Detailed Design.

3.1 Class Diagram

4 Interaction Diagram.

5. Issues, Risk and Dependencies.

6. Future Consideration

7. References.

# **Introduction**

The need for a web application that provides an information about volunteering opportunities that is conducted has widely been entertained. Specifically for the students of Maharishi University of Management where it gives a resource they can use for engaging in events that can boost their resumes and broadening their mind by participating in an extracurricular activity such as volunteering. This project provides a platform students which can update themselves with volunteering opportunities going around the the University and to interact with it by sending comments or posting events they have in mind.

In this task the flexibility of the Spring MVC framework has been utilized. Technologies such as: **JPA, jQuery, Ajax, JSP , Servlets and Restful web-services, tiles** were used in providing a fast and interactive codes. Design patterns such as **singleton, factory methods and PRG** patterns has been implemented. The process of security based on **bcrypt(for encrypting password), validation, interceptors, data-binding, authentication and authorization** also was integrated to build a more secure and flexible system. In this design best design practices has also been followed that can make the system readable, durable and maintainable for future use or extension.

## **1.1** **Purpose**

The project is intended to provide information about volunteering opportunities specifically for the community of MUM. It will provide for an up-to-date volunteering options to be released on timely basis. Information about the volunteering opportunity including its location, organizers, available seats, duration, and intent will be provided. This way a user will find an easy way to sign up for volunteering opportunities on going which is going to help society and makes society more prosperous . Furthermore, users can interact with the released information in terms of posting comments, sharing with friends and collaborating with the system in sharing volunteering stories and blogging about their experience in the platform.

# **2.** **Requirements - Use-Case – Usage Scenarios**

The System requirements are not limited to but are intended to provide the following functionality:

* Log in page: Will provide an opportunity for the user to browse between opportunities and volunteering stories. Moreover will allow for the user to sign in if he/she is a member or join in if not a member.

* Sign up: in this process the user will provide his/her basic information. The person is also given an opportunity to choose from a set of interests for future mapping of volunteering opportunities with the person’s interests.

* Sign in: once signed in the person is provided two types of access based on the status assigned earlier.

If the user is general user than he has following functionalities:

- Opportunities available: a list of volunteering opportunities on one side and a search mechanism on the other side which can help in filtering tasks in terms of category.

- Can be provide interest on particular event.

- Can comment on events,

- Can view detail about the events for further detail.

- If the user wishes to read more about the opportunities a link will be provided that displays another page with full description of the task required and asking if the user wants to proceed further with that particular opportunity.

If the user is an administrator in addition to the above functionality the following links are available:

- A link for registering volunteering tasks and events.

- Can add events category.

- Can approve the user based on seat available and profile.

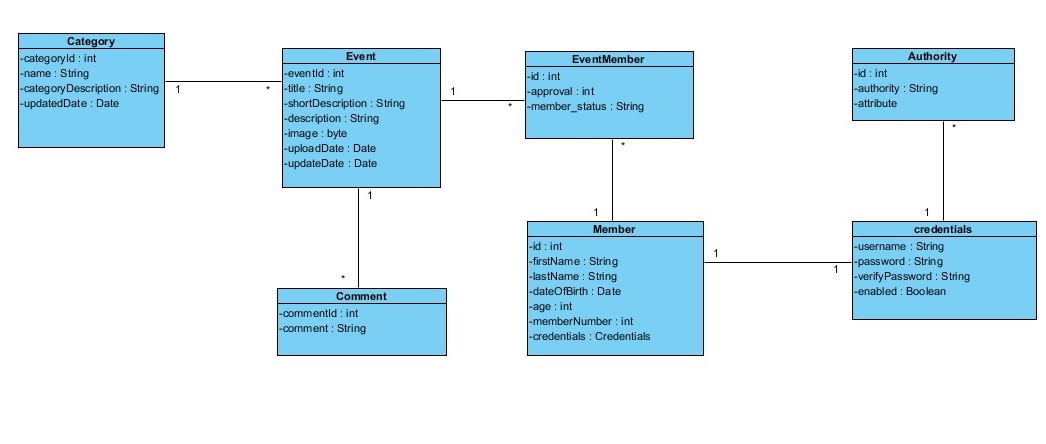
- A link for accessing submitted volunteering opportunities and comments.

Functionality not included but with a prospect of being implemented include:

* Once the user provides his/her confirmation of being part of the volunteering opportunity an email will be sent that gives full description of the opportunity.
* At a time close to the volunteering day an Administrator can send notification to the volunteers reminding them of the upcoming opportunities and updating on any change that might have occurred in the mean-time.
* Functionality can also be added for the administrator to send emails to members that matches their interests.
* The user profile: which provides basic information including the volunteering opportunities the users was part of in the past.
* Opportunities available: a list of volunteering opportunities on one side and a search mechanism on the other side which can help in filtering tasks in terms of date and keywords.
* Volunteering Stories: where the user can read about people’s stories regarding their volunteering experience.

**3**. **Class Diagram**

Below given is the Class diagram of the Project “Volunteer for change”. All together there are eight domain classes, named Category, Event, EventMember, Member, Authority, Credentials, Comment in this project. This Data model is passed to the view from the controller. Category and Event has one to many relation. This gives facility to add the functionality that there can be number of events inside a category. Event and Comment has one to many relation, i.e for each Event there can be multiple comments. The many to many relation between Event and Member is managed by an association class in between Event and Member class. Here EventMember is entity which acts as join table as well as holds the additional data. Entity class Authority provides the roles to the different users. Credentials is used to hold the username as well as password .



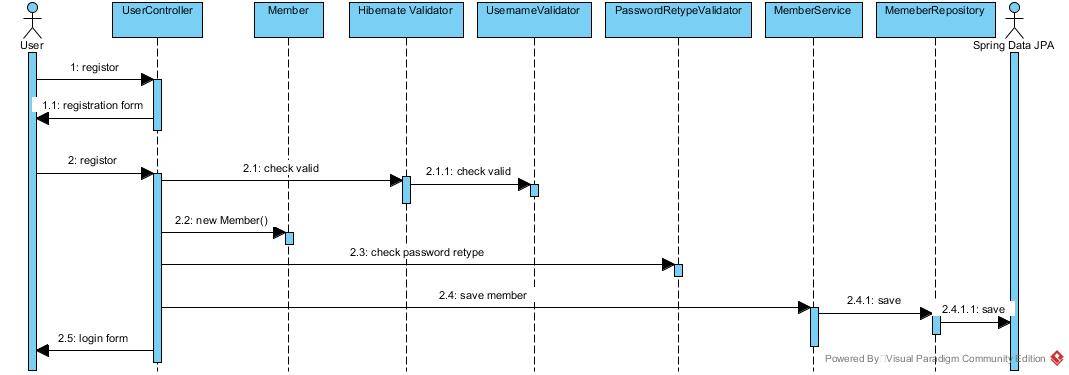
*Fig: Class diagram VFC*

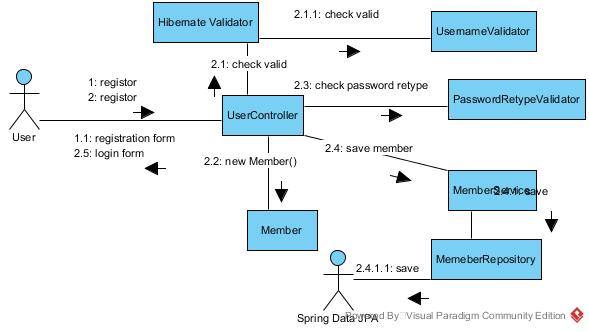
**4** **Interaction Diagram**

**Use Cases:**

**User Registration:**

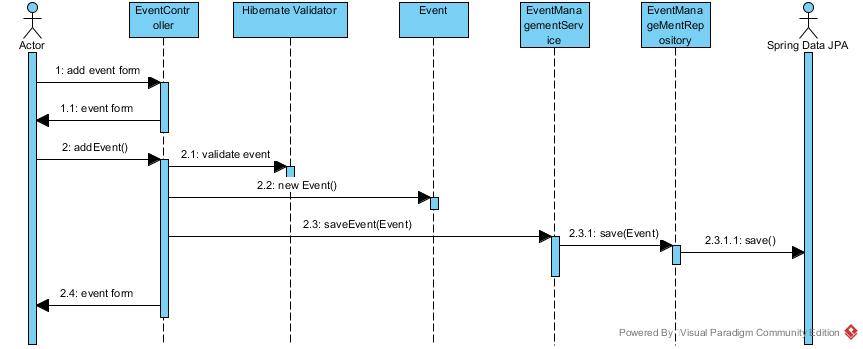
User make request for registration, User controller return the registration form. After filling the registration form, user hit the submit button, User controller redirect the control to the Hibernate Controller, where Hibernate controller validate the data filled by the user. After that Username Validator validate whether the requested user name is available or not. After this stage, re-entered password is matched with the initially entered password. Finally after passing these stages data is saved by using Spring Data Jpa.





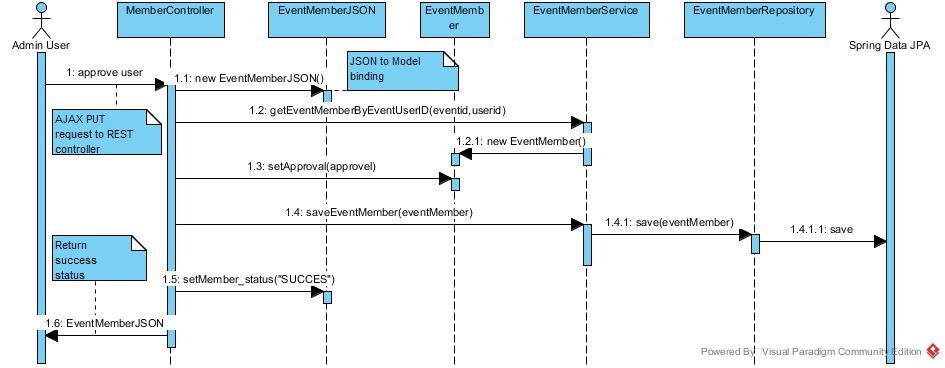
*Fig: Communication Diagram for registration*

**Use Case: Add Event**



Admin make request to add event, Event controller return the add event form, Admin fill up all the data regarding even and submit it, after which the data is validated by hibernate validator. After that event is saved by the controller to the database using EventManagement Service and EventManagementRepository. Now again the event form is returned to add other event.

**Use case: User Enrollment Approval:**



Admin hit the approve button, member controller calls the getEventMemberByEventUserId(eventId, userId) which inturn return the EventMember object where user approval field resides. Now member controller set value to that field and save it to the database.

# **5.** **Issues, Risk and Dependencies**

The problems and issues that we faced and that we deal with has been listed below :

1. **Lazy Initialization:** lazy initialization due to non-used Entity in the another entity which we solved using fetch= FetchType.Eager and removing the whole non-used Entity.
2. **Model Class design:** we had Model Class design where we need to add an extra entity rather than join table since we need to have extra column.
3. **Json Conversion:** There was a problem regarding the JSON conversion which we solved using @JsonIgnore.
4. **Git synchronization:** We had the problem using git, whenever user where trying to merge the project, some data used to be lost. Later this problem was solved using branches in our git repository.
5. **Using tiles:** Since there was missing well structure tile definition a block of <div> was not been able to see.

# **6.** **Future Consideration**

# Initially, we targeted this project for deploying to online user of MUM, but due to lag of time, some of the following features remain for future consideration, which follows:

# **More Secure:** In our project, we have used spring security along with spring security tag library. We have configured the spring security in security config file for the authorization and authentication. As we have only two different roles, we used intercept URL filtering request path for admin and another general user authentication. Most of the part is missing with method level filtering but some of the places we have configured for method level authorization. Still, due to lack of time, we are missing some essential validation for production level hosting.

# **Interactive and good design:** The web application also needs a good interaction and design for user interest and attraction. We have used bootstrap but most of the places we are missing the good design. We can make this application interactive with using more bootstrap features and good CSS with mobile friendly design.

# **Code cleaning:** Code need to be clean and should be easy to understanding to other people too. We have tried to make our code clean and understandable, but we are lagging comment and other documents for giving to another hand. We also put this to our future task.

# **Features :** As illustrated, this project is inspired by developing up and running web application capturing all aspect of the opportunities seeker and needy people. We come up with only a few features but we are a lot to do at the production level. We have completed most of the essential (Event manage, User manager, User registration, User profile view, Admin section), but for the production level most of the needy features like, different user roles, user can create event and share, comment option, admin shorting and interest candidate sorting and filtration, email notification, user forget message email setup etc.

**7.** **References**

1. Spring MVC Beginner’s Guide, June 2014.

# 2. Web Application Architecture (WAA), Lectures and Demo, Maharishi University of Management.

# 3. Spring MVC Web framework reference http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html