Project Topic:

Event Management for social service website using Ruby on Rails: BTOEventView

Short description of project work:

The application is intended for Bhutan Toilet Organization (BTO), a non-profitable organization located in Bhutan, which provides clean toilet facilities during public events organized. Currently, BTO is using the manual paper-based system in managing and facilitating the events and activities organized by them and their clubs who contribute to achieving the mission of BTO. By having the Web-based Event Management System (BTOEventView), it will let the authorized user create and update the activities or events and the users and public will be more alert and aware of the existence of the events or activities held in BTO.

For our project, there is only one major interface that will be displaying the events held or in progress by the organization to the public.
Therefore, the Project Manager, College Club Ambassadors, and Admin is allowed to do the CRUD operations and others can only view the Information.

Our web-based system or application – BTOEventView

1. Will display the Events of all the activities organized by the Bhutan Toilet Organization and its associations (college clubs).
2. Will be displaying events details from the event organizer.
3. Will be displaying the event's organizer.
4. Will be displaying the location of events.

Design details of the content data model with potential users of the system.

In general, there is user-submitted content with authoring workflow specific to our project.

Admin

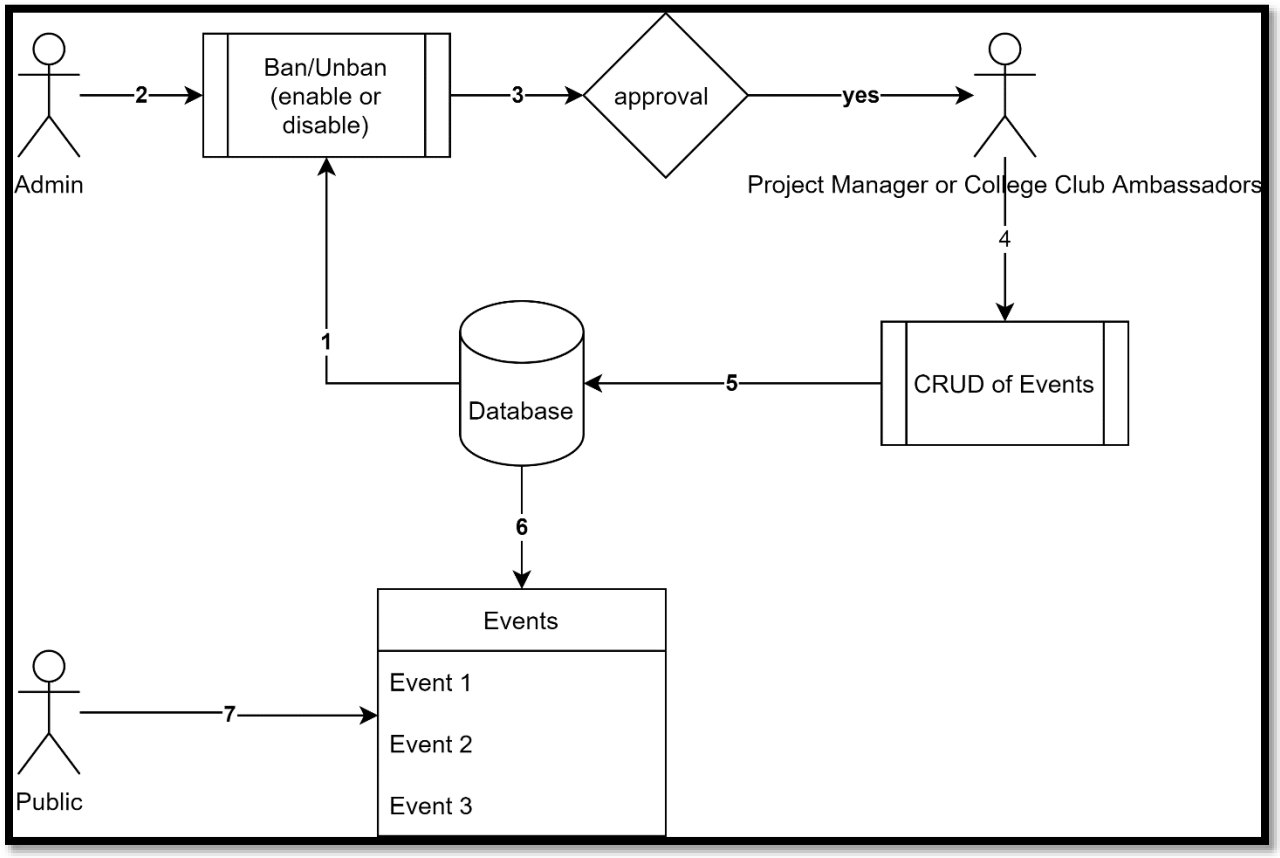
- He/she can ban/unban users, view user status and perform all tasks.

Project Manager and College Club Ambassadors

- The above user can perform CRUD operations i.e., create events, read events, update the events, and delete the events.

Public

- all can view the ongoing and past events.



The above figure content data model is managed as:

1. User registration is retrieved from the database for admin.
2. Admin can check the list of users registered and perform the action to enable and disable.
3. If the user is approved by the admin, then the user becomes a project manager or college club ambassador.
4. Once approved, the user can perform a CRUD operation for events.
5. Then actions are stored in the database.
6. The main interface retrieved the information about the event from the database.
7. The general user or public can view the list of events created by the approved user.

Workflow design of the application

It is a very fundamental project where we planned on making our website pretty simple. As our project will be displaying the Events organized by the BTO. We planned on having the link to this

along with their website. On clicking the link, the users can view the events in a table format with their organizer and below it, we will be displaying all details that are required to be displayed.
We have only one interface where the operations can be performed by the Project Manager and College Club Ambassadors. The other workflow of our design is explained in the above figure.

Document of the versioning design on your site.

For our BTOEventView application, we are not maintaining any versioning system separately. During development, GitLab takes care of the version control.
For Versioning concept used for the concurrent user trying to update events in our application:

- ➔ Optimistic locking is used where we allow concurrent users (Project Manager and College Club Ambassador) to perform update events they like but track updates to the database.

When one user attempts to update an old version of a record, an exception and transaction rollback occurs.
In Rails, optimistic locking is enabled on any ActiveRecord class by adding a version column to the database table:
alter table events add column lock_version int default 0;

UATs to pass. Users should be able to get something up on your server and see it published.

```
Feature: Test PS4

  Scenario: check user management model           # features/ps4.feature:3
    When I visit the main page                    # features/step_definitions/ps4_steps.rb:1
    Then I should see a ps4 index link            # features/step_definitions/ps4_steps.rb:5
    When I click on the ps4 link                  # features/step_definitions/ps4_steps.rb:9
    Then I should see Define your user data model # features/step_definitions/ps4_steps.rb:13
    When I see User Management doc link           # features/step_definitions/ps4_steps.rb:17
    Then I close                                  # features/step_definitions/ps4_steps.rb:21

4 scenarios (4 passed)
38 steps (38 passed)
0m1.725s

  Share your Cucumber Report with your team at https://reports.cucumber.io

  Command line option:    --publish
  Environment variable:   CUCUMBER_PUBLISH_ENABLED=true
  cucumber.yml:           default: --publish

  More information at https://cucumber.io/docs/cucumber/environment-variables/

  To disable this message, specify CUCUMBER_PUBLISH_QUIET=true or use the
  --publish-quiet option. You can also add this to your cucumber.yml:
  default: --publish-quiet
```

All social Web applications have users, content, and user-to-content map. The following figure will explain the functional requirement.

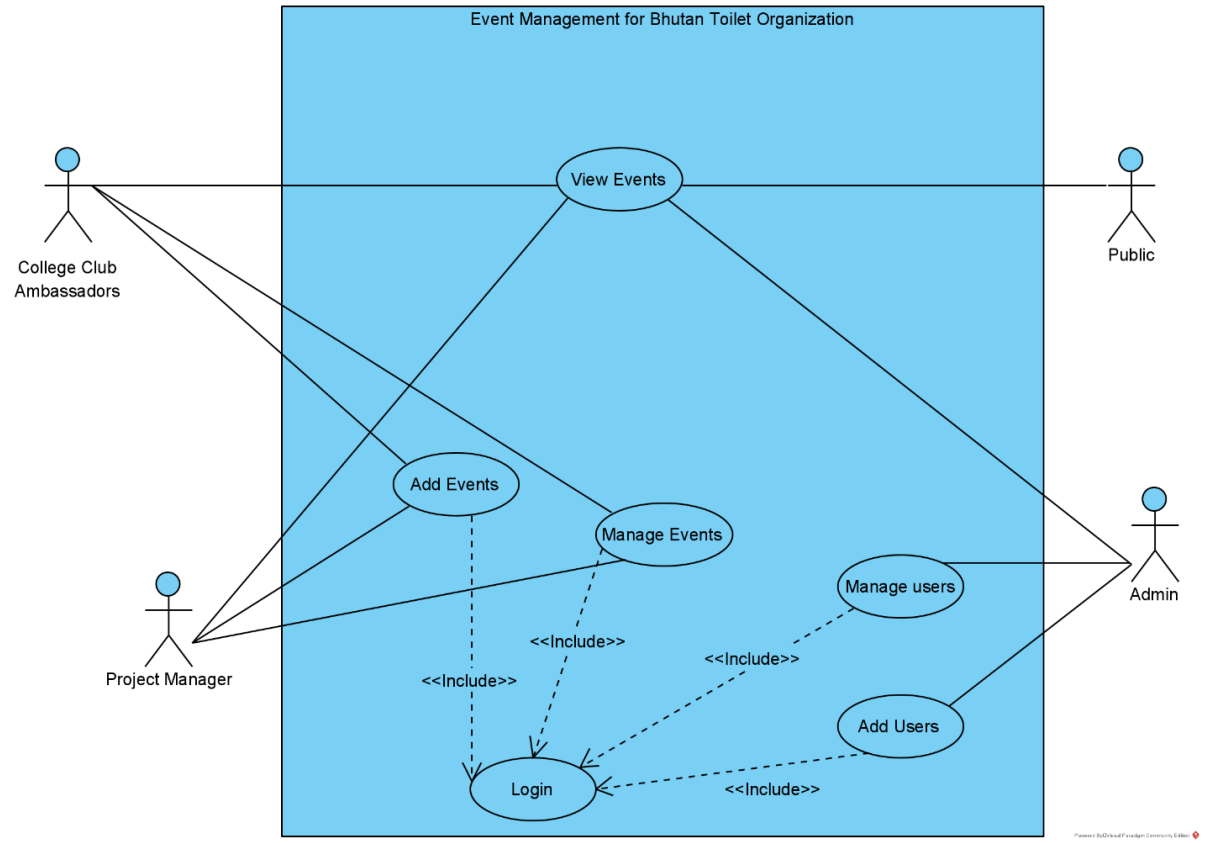


Figure 1. Use Case diagram

As a General Public will be able to:

- view the ongoing events
- view the past events

Project Manager and College Club Ambassadors will be able to:

- add events
- manage events (update and delete)
- update user profile

Admin will be able to:

- register user into the system
- perform all the action
- add users and add permission

- can disable and enable users.

The method used for developing an application:

1. Multi-page applications (MPAs) and Single-page applications (SPAs).

For MPAs we will focus on the software architecture and technology used on the server-side.
For SPAs, we look at the architectures of both the frontend and backend systems and their interactions.

2. Rapid Application Development (RAD) / Prototyping

The main advantage is development application has resulted in less rejection when the application is placed into production since development is done with the end-user.

Security concerns:

For security reasons and to have a secure application, we have designed and implemented it as if the system was under constant attack.
Therefore, we maintain the state over a series of response/request interactions

- hiding state information in documents delivered to the user,
- storing information about user sessions on the server and
- responding to user events client-side, without starting a new HTTP request/response cycle.

Avoid SQL Injection

SQL injection mainly works when the attacker can insert his/her query which would be SQL keywords that are directly executed in raw SQL query the backend.
Then the attacker can access numerous data and update his/her privileges. To avoid this, the input taken from users is not directly used in the SQL query. We would rather use parameterized queries and dynamic attribute-based finders.
Example:

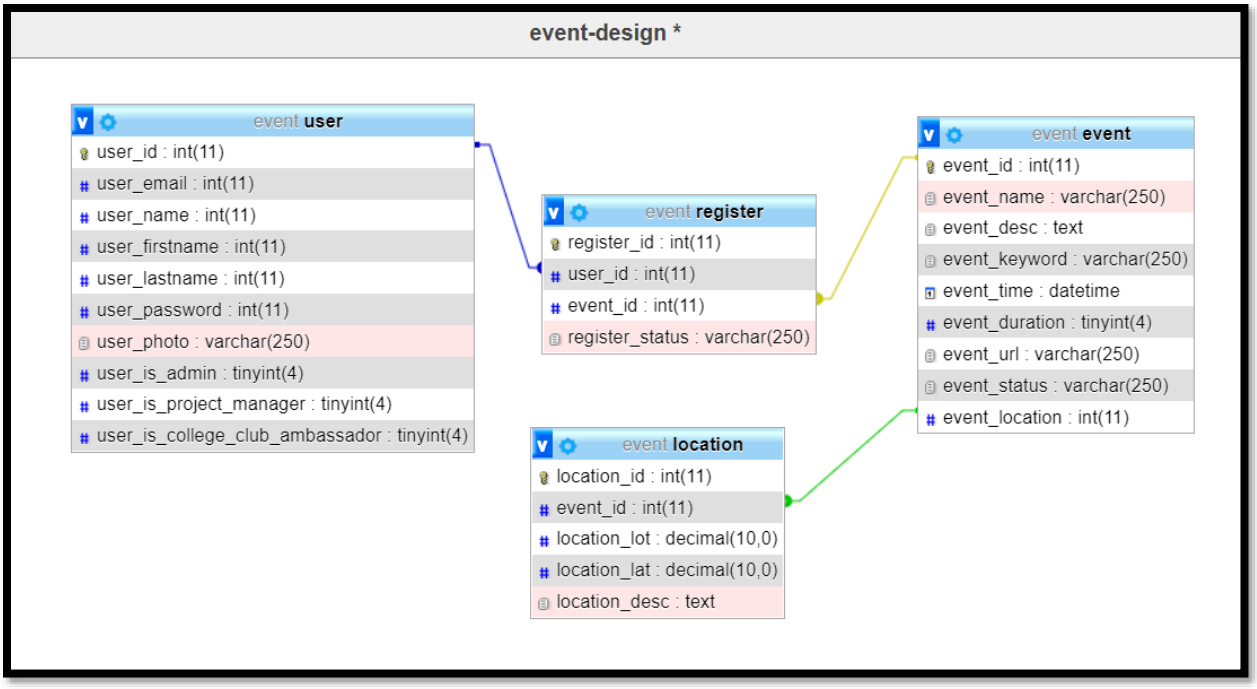
```
User.where("name = '#{params[:name]}'") # SQL Injection!
=>
User.where(["name = ?", "#{params[:name]}"]) # No SQL Injection
```

Avoid XSS attacks

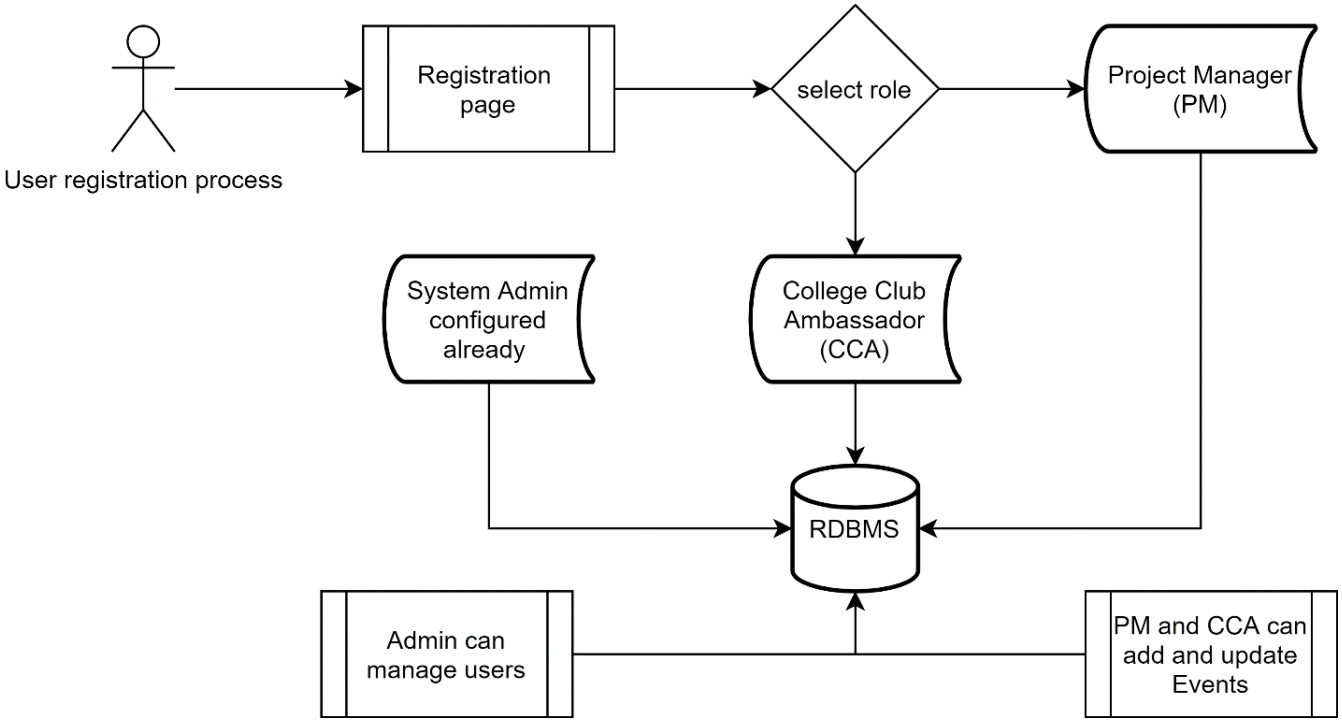
XSS attacks are Cross-Site Scripting attacks that occur when an attacker can send malicious code, usually in the form of a browser side script, to a different end-user.
Rails have a built-in XSS protection mechanism which automatically HTML escapes all the data being transferred from Rails to HTML.
HTML escaping substitutes HTML entities such as '<' and '>' with '& lt' and '& gt' so that the scripts "<script>" "</script>" will be escaped. Hence, whatever malicious code the attacker may post to the application will be HTML escaped and not get executed.

Define your user data model.

Referential integrity is the relationship between different models/tables. Rails active record provides dynamic finder methods that are simple to use and fast to execute.
The database consists of 4 tables shown in the figure. The user table represents the users of BTOEventView. It maintains 3 different roles i.e., admin, project manager, and college club ambassador. It has other info first_name, middle_name, and last_name. It also has is_XXX which is used to ban/unban user account. Event has event name, description, time, duration. It has a foreign key that is a user id from the user table. Events have a many-to-many relationship with the User table through the Register table. That is, a user has many events, and an event has many users. Register table is a join table for user and events. It joins them using uid (users) and event_id (Event). Location table is used to store the location description.



User registration and management page flow:



SSL enabled for the complete application to prevent password sniffing and make session cookies are HttpOnly:

- User registration and login are implemented using the 'devise' gem in rails. We have further email confirmation when a new user wants to register.
- Users can use the forgotten password to get a reset token to his/her mail.
- We added SSL for the website and forwarded HTTP requests permanently to HTTPS. Hence, all the interactions will be authenticated.

Finally, some significant suggestions for making testing job to be effective for the development of application could be – Test Early - Test Often - Test Automatically. Thus, we can work on the following:

- Get a production build up and running on a test server early in the development and give stakeholders access to the server.
- Create an automated regression test suite and keep adding to it as defects are found.
- Use a continuous integration tool such as Jenkins or GitLab CI to automatically download the latest build, run the test suite, and generate quality metric reports.
- Add appropriate tests as you code and ensure the regression test suite passes before every commit.