CSC517 – Object Oriented Languages and Systems

Project report

Date/Time Functionality

Team

AmitMendiratta(amendir2@ncsu.edu)

NabajyotiPatowary(npatowa@ncsu.edu)

AlokSarang(avsarang@ncsu.edu)

ShayanSinha(ssinha3@ncsu.edu)

Contents

REVISION HISTORY:

NAME	DATE	CHANGES	VERSION
Fall 2012_Team3	11/01/2012	INITIAL DESIGN DOCUMENT	1.0

1. INTRODUCTION

1.1 Problem Definition and Description:

The following features with respect to date/time need to be investigated and developed/rectified in Expertiza.

1. Currently the time format that is used to display time in Expertiza is UTC format which creates confusion for users located in different time zones. The aim of this project is to allow Expertiza to handle multiple time zones.
2. A user's profile should give that user's preferred time zone. When a user (e.g., an instructor) creates other users, the other users by default should have the same preferred time zone as the user who created that user. However, all times in Expertiza should be displayed in the user's preferred time zone which he can save in the User's Profile page.
3. Expertiza should know about daylight savings time (DST) and should be able to adjust the time when DST begins and ends.

2. Requirements

The requirements can be listed as follows:

1. By default instructors time zone shall be set as the user's default preferred time zone.
2. If the user has specified a time zone then that shall be set as the users default time zone.
3. All the places where time is displayed like assignment submission time, review time etc. should display the time in user's preferred time zone.
4. The software should also be able to take care of Day Light Savings if that is observed in the location where Expertiza is running.

3. DESIGN

3.1 Modules:

Users Model : The primary change that would take place in the users model as a new field called timezonepref has to be added. This by all means will be a string data type.

The modified user's model shall look like this-

```
"name" :string
"password" :string
"role_id" :integer
"password_salt" :string
"fullname" :string
"email" :string
"parent_id" :integer
"private_by_default" :boolean
"mru_directory_path" :string
"email_on_review" :boolean
"email_on_submission" :boolean
"email_on_review_of_review" :boolean
"is_new_user" :boolean
"master_permission_granted" :Boolean
"timezonepref" :string
```

Fig 1: User's modified table

Code changes

New method created for time zone. It keeps the old time zone in a temporary variable.
If the user is not logged in then the default time zone is set as per the migration.
Once the user logs in, his/her preferred time zone is set by retrieving the preferred time zone from the table using the current user session.

expertiza/app/controllers/application_controller.rb

```
def set_time_zone
  old_time_zone = Time.zone
  logger.debug !(session[:user].nil?)
  if (!(session[:user].nil?))
  current_user_id = session[:user].id
  preferredtimezone  = User.find_by_id(current_user_id).timezonepref
  logger.debug preferredtimezone
```

```
  Time.zone = preferredtimezone.to_s if logged_in?
  end
 ensure
  yield
  Time.zone = preferredtimezone
 end
```

```
def create
  @current_user_id = session[:user]
  # if the user name already exists, register the user by email address
  check = User.find_by_name(params[:user][:name])

  # set the current user id to the admin's user id so the new users created will have the same prefered
time zone as admins by default.
  @user.timezonepref = @current_user_id.timezonepref
End
```

File :- expertiza/app/views/profile/edit.html.erb

(New Dropdown added in profile page of users to set preferred time zone)

```
 <p><label >Preferred Time Zone</label>

 <%= time_zone_select(:user, :timezonepref) %> <br>
```

---

4. USE CASE DIAGRAM

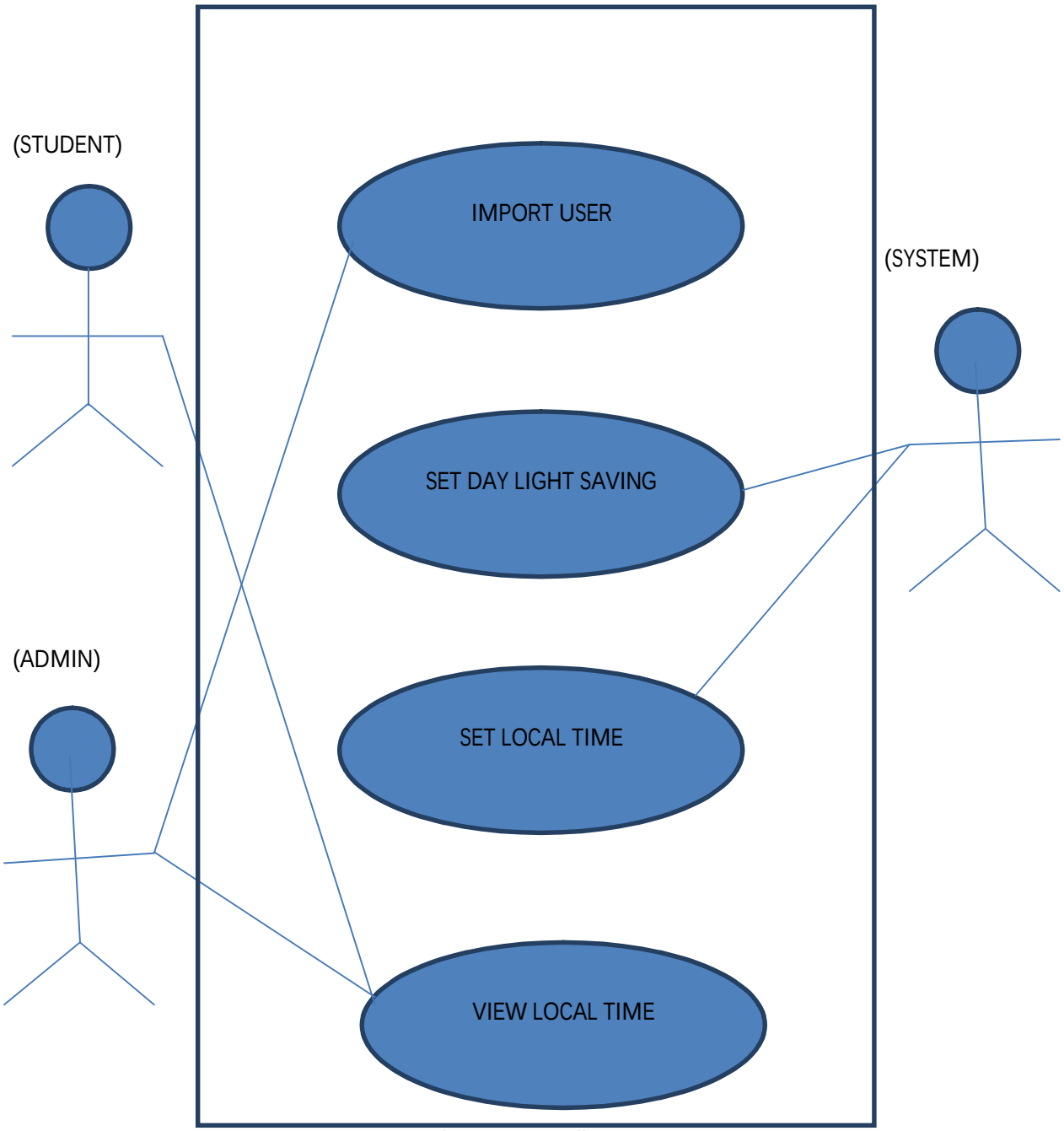

Fig 2: Use case diagram

Actor	Description
Admin	This actor is responsible for adding new users.
Student	This actor is responsible for viewing his assignments along with time in local time format.
System	This actor is responsible for automatically setting the local time accommodating day light saving.

Name: Admin adding new users to the system.

Actor: Admin

Other Participants: System

Precondition:

1. Admin should be logged in.

Primary Sequence:

1. Log in to Expertiza as Admin/Instructor

2. Select Manage

3. Select create new user.

4. Confirm and Submit.

Post Condition:-

1. New User created with default time zone set as Admin/Instructor's (who created him) time zone.

Actor	Description
Student	This actor is responsible for Setting his preferred tome zone.
System	This actor is responsible for automatically setting the local time accommodating day light saving.

Name: New User sets his preferred time zone.

Actor: New User

Other Participants: System

Precondition:

1. New User should be logged in.

Primary Sequence:

1. Log in to Expertiza with Username and password

2. Select Profile

3. Select preferred time zone.

4. Click Save

Post Condition:-

1. User's preferred time zone get changed to newly selected timezone.

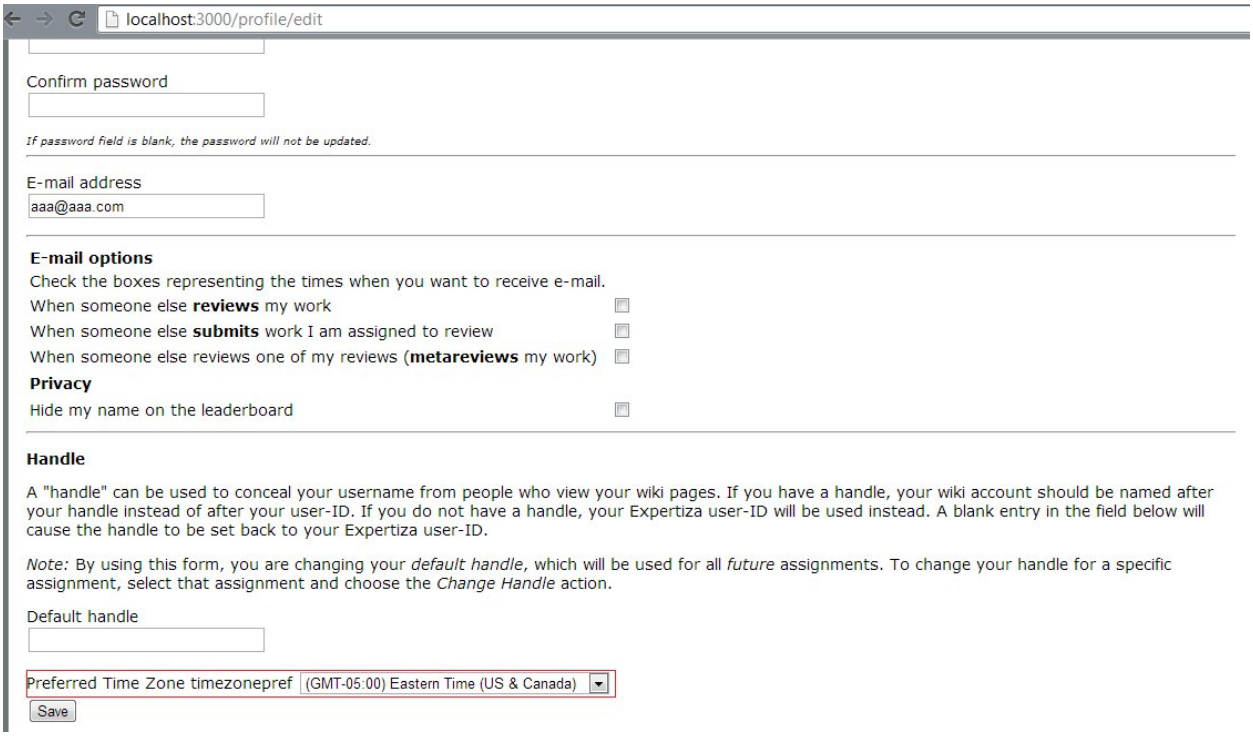

Fig 3: Profile

Name: System incorporating DLS (Day Light Saving) to calculate and set the current local time.

Actor: System

Other Participants: NA

Precondition:

1. Database containing records of new users to be added along with their preferred timezone with appropriate delimiter.

Primary Sequence:

1. System sets the current local time as per the DSL (Day Light Saving) calculated.

5. Object Oriented Design

As with most Ruby on Rails applications, this solution uses the Model-View-Controller design pattern. The following details shows a combination of new and existing classes along with the attributes and operations that are relevant to the implementation of this solution.

User Model - Updated with new Column named timezonepref

Application controller – New Function added to set preferred time zone.

View – Update to incorporate changes for a new drop down for preferred time zone.

6. DESIGN Principles to be followed

1. MVC – The project is implemented in Ruby on Rails that uses MVC architecture. It separates an application's data model, user interface, and control logic into three distinct components (model, view and controller, respectively).

2. Dry Principle – We are trying to reuse the existing functionalities in Expertiza, thus avoiding code duplication. Whenever possible, code modification based on the existing classes, controllers, or tables will be done instead of creating the new one.

7. Limitations

1. If sometime later a new requirement comes up to add another feature for user then another new column has to be added in the user's model, this would become cumbersome for every new feature added for the user. So it would be better to have a Preferences table to include all the user preferences.