```
<?php

namespace Drupal\student_enrollment\Form;

use DateTime;
use Drupal;
use Drupal\Core\Form\FormBase;
use Drupal\Core\Form\FormStateInterface;
use Drupal\node\Entity\Node;
use Symfony\Component\HttpKernel\Exception\AccessDeniedHttpException;

/**
 * Provides a Student Enrollment form.
 */
class StudentEnrollmentForm extends FormBase {

  /**
   * {@inheritdoc}
   */
  public function getFormId() {
    return 'student_enrollment_student_enrollment';
  }

  /**
   * {@inheritdoc}
   */
  public function buildForm(array $form, FormStateInterface $form_state, $course_id
= NULL) {

    // Prevents administrators from enrolling for a course. They are redirected to 4
03 page(access denied).
    $current_user = Drupal::currentUser();
    if (in_array('administrator', $current_user->getRoles())) {
      throw new AccessDeniedHttpException(); /* throws new exception for denied acce
ss to the current page.
      The 403 page is displaying.*/
    }

    // Build the enrollment form elements here.
    $course_name = Node::load($course_id)->label();
    $form['course'] = [
      '#type' => 'label',
      '#title' => '<h2>' . $this->t('Enroll for the course') . " " . $course_name .
'</h2>',
    ];
    $form['course_id'] = [
      '#type' => 'hidden',
      '#value' => $course_id
    ];
    $form['submit'] = [
      '#type' => 'submit',
      '#value' => $this->t('Enroll'),
    ];

    return $form;
  }
  /**
   * {@inheritdoc}
   */
  public function validateForm(array &$form, FormStateInterface $form_state) {
    // Implement validation logic if needed.
  }

  /**
   * {@inheritdoc}
   */
  public function submitForm(array &$form, FormStateInterface $form_state) {
    // Implement submission logic to record the enrollment in the database.
    $user_id = \Drupal::currentUser()->id();
    $course_id = $form_state->getValue('course_id');
    $course_start_date = new DateTime(Node::load($course_id)->get('field_start_date'
)->value);
    $now_date = new DateTime();
    // Check if the user is already enrolled in the selected course.
    if (!$this->isUserEnrolled($user_id, $course_id)) {
      if ($now_date > $course_start_date) {
        // Record the enrollment.
        $this->recordEnrollment($user_id, $course_id);
        $this->messenger()->addMessage($this->t('Enrollment successfull.'));
        //$this->notificationService->sendEnrollmentNotification($user_id, $course_i
d); // send an email for successfull enrollment.
      } else {
        $this->messenger()->addMessage($this->t("Enrollment time for this course has
 expired because it is after the start date."), 'warning');
      }
    } else {
      $this->messenger()->addMessage($this->t('You are already enrolled for this cou
rse.'), 'warning');
    }
  }

  /**
   * Check if a user is already enrolled for a course.
   *
   * @param int $uid
   *   The user ID.
   * @param int $course_id
   *   The course ID.
   *
   * @return bool
   *   TRUE if the user is already enrolled, FALSE otherwise.
   */
  private function isUserEnrolled($uid, $course_id) {
    // Implement logic to check if the user is already enrolled for the course.
    $query = \Drupal::database()->select('student_enrollments', 'e');
    $query->fields('e');
    $query->condition('user_id', $uid);
    $query->condition('course_id', $course_id);
    $result = $query->execute();

    return !empty($result->fetchAssoc());
  }

  /**
   * Record a user's enrollment in a course.
   *
   * @param int $uid
   *   The user ID.
   * @param int $course_id
   *   The course ID.
   */
  private function recordEnrollment($user_id, $course_id) {
    // Implement logic to record the user's enrollment in the database.
    \Drupal::database()->insert('student_enrollments')
      ->fields([
        'user_id' => $user_id,
        'course_id' => $course_id,
        'created' =>  \Drupal::time()->getRequestTime()
      ])
      ->execute();
  }
}
```