****

**UNIVERSITY OF NEW YORK TIRANA**

**M.Sc. in Computer Science**

Coursework in: Engineering Advanced Web Applications Coursework

**Kristi Mita**

1. **Introduction and problem description**

This project is about developing a Saas application to help a university instructor to manage the courses he teaches and to provide the students with an easy way to check their grades. This application is developed under the assumption that only one instructor will use the application, but it can be scaled for more users. The application has 2 type of users, the instructor and the student. The instructor has more privileges than a student. Also everyone can see a welcome page and without logging in can see a list of all courses and for each course the minimum, maximum and average grade .

An instructor once logged in can create, read, update and edit students accounts, courses, assessment items and grades. It can assign a student to a particulary course, can add assessment for each course and assign grades for each assessment.

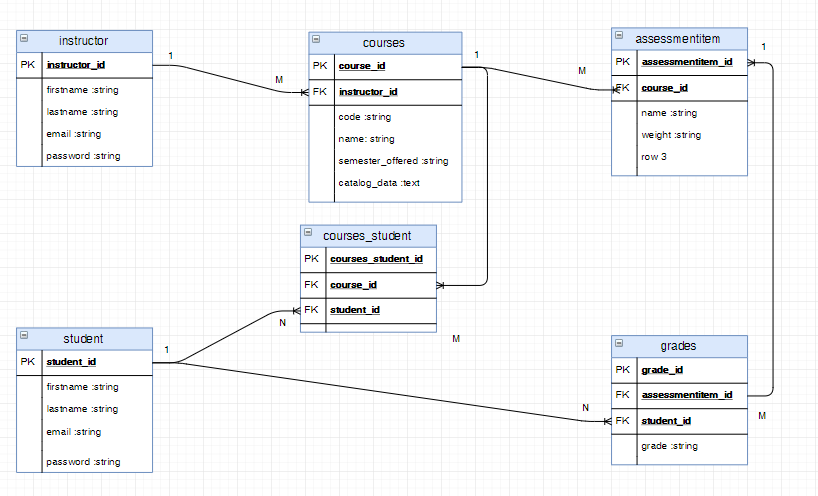
The user stories for this application were:

* The application has a welcome page (/welcome) which handles public functionality
* Whenever not logged in, any access to other views ( by typing the corresponding URL) should redirect to user login page
* After logging in, the instructor should be redirected to instructor’s main page.
* When in instructor’s main page, the instructor should be able to go to (a) personal info view (b) the courses index view, (c) students index view, or (d) back to welcome view.
* In the personal info view, the instructor should be able to change name, surname, email, or password
* From the students index view, the instructor should be able to perform all the CRUD functionality for student resource.
* From the courses index view, the instructor should be able to perform all the CRUD functionality for course resource.
* Additionally, course view must also contain a link to index view of the students registered on that course.
* From the index view of the students registered on that course, the instructor should be able to do the CRD functionality (list all students of a course, add a student to the course, delete students from the course but nothing to update on the student-course level). An appropriate hyperlink on the student level should take the user to index view of student’s grades for that course.
* From the grades index view, the instructor should be able to list, add, delete, update grades of one particular student for the chosen course.
* After logging in, a student should be redirected to the student’s main page.
* In the student main page, all courses that the student is registered with are shown, and for each course, the computed overall grade is listed.
* When clicked on an appropriate link, the list of grades that the student has taken on that course will be shown.
* The student cannot make any modifications to the grades.
* When logging out, all users will be redirected to the welcome page.

Also, some validations were required to prevent users from entering non sense information:

* For instructor and students
  + name, surname, email and password must be present
  + name and surname are maximum 30 characters each, the concatenation of two must not exceed 50 characters
  + email must be validated with an appropriate regular expression
  + password must be at least 8 characters long, must contain at least one uppercase letter, one lowercase letter, and one numerical character
  + email must be unique over all students + the instructor
* For courses
  + code, name and semester of the course must be present
  + code must be 5 or 6 characters long
  + semester must be at most 10 characters long
  + catalog data must be at most 500 characters long
* For assessment items
  + and weight must be present
  + title must be at most 30 characters long
  + weight is a floating point between 0 and 1
* For assessment grades
  + Grade must be present
  + grade is an integer between 0 and 120

1. **Database Design**



For this application six tables are used. In the above db schema image the instructor has a one to many relationship with the courses. Courses and students have a many to many relationship which is expressed by another table which is courses\_student. The instructor has a one to many relationship with students thorught courses. A course can have many assessment items.A student can have many grades which is a one to many relationship for different assessment items. A grade belongs to a student for a particular assessment item. From the assessments item ID we can find the course and than the instructor of that course.

1. **Implementation**

To create a rails application type the command rails new application\_name (example: gradekeeping\_system)

1. **Routes**

Below is the config file for the routes that have been used in this application

resources :courses do

resources :assessmentitems

end

resources :instructors

resources :students

root :to => redirect('/welcome')

get '/welcome', to: 'sessions#welcome' # it is the welcome page, this is showed to the user when it opens the app

get '/instructor\_home', to: 'sessions#instructor\_home' # this is the instructor homepage

get '/student\_home', to: 'sessions#student\_home' # this is the instructor homepage

get '/login', to: 'sessions#new' # show login page

post '/login', to: 'sessions#create' # when suer clicks login button (submit form)

post '/logout', to: 'sessions#destroy' # logout

get '/courses/:course\_id/index\_students' => 'courses#index\_students' # show all students for a course

get '/courses/:course\_id/add\_student' => 'courses#add\_student' # add a studen to a course

get '/courses/:course\_id/remove\_student' => 'courses#remove\_student' #remove student from course

# below are the path to perform all CRUD operation os grades

get '/courses/:course\_id/student/:student\_id/grades' => 'grades#index'

get '/courses/:course\_id/student/:student\_id/grades/new' => 'grades#new'

post '/courses/:course\_id/student/:student\_id/grades' => 'grades#create'

get '/courses/:course\_id/student/:student\_id/grades/:id/edit' => 'grades#edit'

put '/courses/:course\_id/student/:student\_id/grades/:id' => 'grades#update'

delete '/courses/:course\_id/student/:student\_id/grades/:id' => 'grades#destroy'

In this application the instructor, the courses and the student are completely separated by each other. Only the courses and assessment items are nested together. Also the welcome page and the session routes are defined. The grade routes are also defined manually by using RESTful principles. There is only one login page used both for instructor and for student from which the user has to choose form a dropdown if it will be looged in as student on instructor.

1. **Create database tables**

For each database table a migration file is generated using the command rails generate migration create\_table\_name. This will create a migration file which will represent the database table. In the migration file we define the colums that we need for the table and then we generate the table by using the command rake db:migrate.

Below is an example of student migration file:

class CreateStudents < ActiveRecord::Migration[5.1]

def change

create\_table :students do |t|

t.string 'firstname'

t.string 'lastname'

t.string 'email'

t.string 'password'

t.timestamps

end

end

end

1. **Models**

This application uses five model classes which are the instructor, student, course, assessment item and grade. This is where the relationship between entities are defined by using rails keyword belong\_to or has\_many etc. Also some validation are also provided for each model and some other helper methods which are needed to calculate the maximum, minimum and average grade of a student. You can find the validation requirements above at section one.

1. **Controllers**

The are six controller classes used by this application. It is the instructor, student, course, assessmentitem, grade and session controller. The session controller is responsible for rendering the welcome page, and the instructor and student homepage. Also it handles the login and logout process. In the default controller application\_controller some helper methods are defined to check if there is any user logged in or not. To create an instance variable (@current\_user) which is been used to keep information about the logged in user. Also some helper method are used to not allow users of type student to use certain features of the system like adding a new student, or editing a grade, etc.

1. **Views**

There is one view for each controller. The views are codded in Haml and in some of the views partials can be found. Also all the notice and warning message are placed in layout/application.html.haml to enfoce dryness. Some custom css style has been used to make the application look more beautiful.

1. **Conclusion**

To run the application we need to type the command rails s and the application will start at localhost port 3000. All the requirements and user stories have been fillfilled. The application is developed on a local machine running Window 10, Ruby 2.3.3p222, Rails 5.1.6.1 and using PostgreeSQL as a database because of some problems with sqlite3 gems. The application has only one instructor which is added manually from PostgreeSQL database interface, or it can be added by using the seed.rb file. There is no interface to add an instructor.But this functionality can be easily added later.