Online Academic Registration Portal
CS252 Course Project

Abhinav Garg, Anuj Nagpal, Nayan Deshmukh

IIT Kanpur

Introduction

OARS is the online academic registration portal of IIT Kanpur. The current platform is very outdated and old-fashioned and it has been causing a lot of problems to entire student community and faculty. Owing to the need of revamping the current website, we attempted to make a new portal on Ruby on Rails framework.

Features

Features of our application in brief:

- Three types of users - Students, Instructors and OARS admin
- Students can access their transcript which includes list of their completed courses as well as ongoing courses.
- Students can also choose among the courses offered next semester and request for them. They can also see details of each course.
- Contrary to Students, each Instructor can see the details of other instructors. Instructors can view pending course requests and accept/reject them.
- OARS admin can access everything. He/She can create, edit or delete any student, instructor or course.
- Each user can update his own personal details or password.

Models in our MVC model

Model	Description
User	For Students + OARS admin
Instructor	For Course Instructors
Course	For Courses Offered in Upcoming Semester
CourseStore	For Ongoing or Already Completed Courses
Request	For Requesting Courses

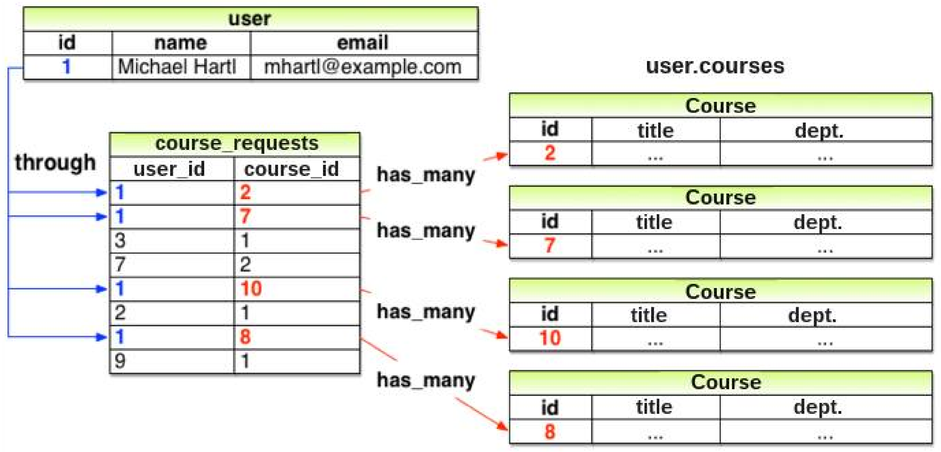

Figure 1: Model Relations in our DataBase

Access and Authorization

- A student can edit/view his/her own personal details, view any course and request for a course but can't edit/view details of any other student or instructor.
- An instructor can edit/view his/her own personal details, view details of any other instructor, accept/reject course requests but can't edit/view details of any student and edit any course or instructor

Any unauthorized access will redirect the url either back to current user details or an error page

Description of Implementation

- **Interfaces** : We have two major interfaces other than the CRUD for each model, i.e. request_course for user model and accept_course for instructor. request_course creates a new request with a user_id and a course_id selected by the user.
- **Seed File** : To generate random data for our database, we have created a seed file which generates many users, courses, instructors, ongoing courses, completed courses and course requests automatically.
- **Cookies** : For our browser to remember the user we have used temporary cookies. Use of Permanent cookie was avoided as oars is accessed only once. Unlike trivial usage of user id as a cookie parameter we have used email to work smoothly across all the models.
- **Friendly Forwarding** : By also storing the requested url in the cookie we have implemented friendly forwarding, Which means that if you request for a page and you get login page then after login you are automatically redirected to the requested page.
- **Admin** : We have added an 'admin' boolean to both user and instructor tables. An admin has all CRUD operation rights apart from all the rights that a user or instructor have.
- **Testing** : We have tested our model through automated testing feature provided by rails. We have written over 10 automated tests to test for bugs and security issues in our app

Model Relations

Each user has a list of courses which were requested by user. This list is obtained through the entries of course_request table. User ID and course ID are the foreign key in requests table. Similarly each course has corresponding user_requests and the list of the user through the user_requests list. Each course model belongs_to a Instructor and each instructor has_many courses. The CourseStore model is to store the ongoing and completed courses of a particular user.

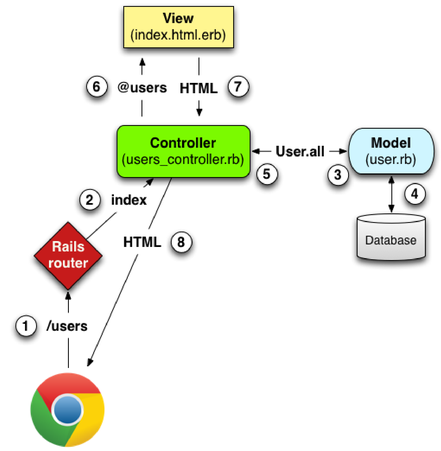

Figure 2: MVC Model

Future Work Possible

- Add/Drop Feature
- HSS Course Lottery
- Form Submission to DUGC
- Improving Scalability and Security

References

- [1] Ruby on Rails Tutorial by Michael Hartl https://www.railstutorial.org/book/
- [2] RailsGuides http://guides.rubyonrails.org/v3.2.9/