CSCI3356 Software Engineering

G.A.L.S. Delivery 4 Report

Group Members:
Sophie Appicelli
Kyle Gil Tan
Andrew Lim
Ashley Sachdeva

Introduction

Throughout the semester, we were tasked with creating a software to handle peer evaluations using Django and SQLite. This hands-on project required the existence of 3 types of users, admin, professors, and students, multiple courses, and dynamic forms for each course. During this course, we learned about the basic lifestyle of software development and how to work in teams, and as a result we applied that newfound knowledge to the creation of this software. Acquiring knowledge of many new technologies such as Django, Git, and Bootstrap, all group members stepped out of their comfort zones and dedicated several hours of research to become familiar with these tools. The project proved to be intellectually challenging at times frustrating, but it also allowed us to gain experience with regards to the duties of software engineers.

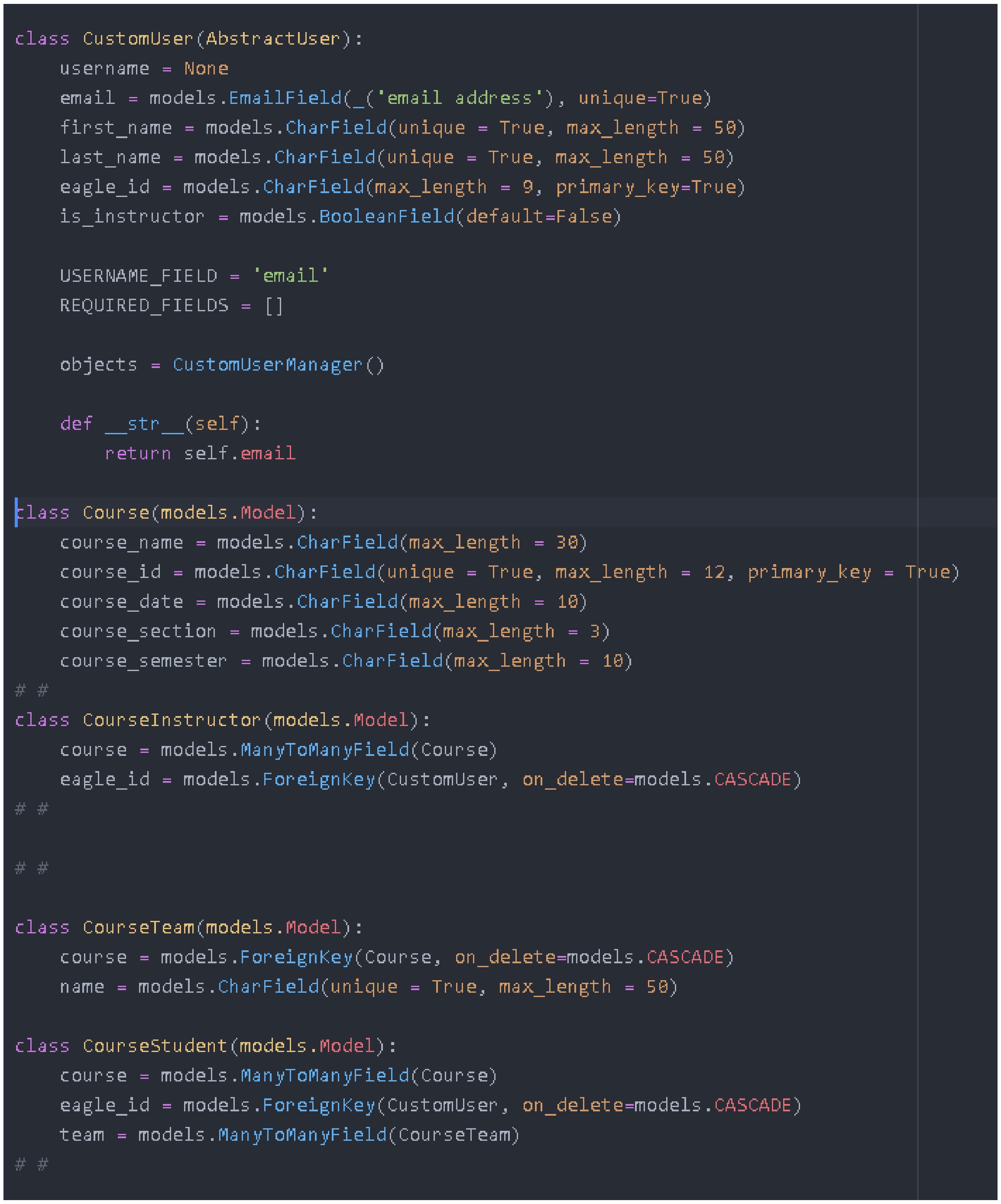
Group Methodology

Our group utilized a combination of individual coding work in our own personal selection of IDEs in addition to frequent group meetings over Zoom in which we would either distribute individual tasks or focus all our combined efforts on one individual task. A Slack server was set up for easy storage of links, code segments, and general discussion, in addition to an iMessage group chat. Full group meetings took place every 2-3 days with additional meetings between smaller group subselections depending on which group members were assigned to tasks. Task assignment was more or less determined collectively with bias toward group member skills (ie: Ashley and Sophie had strengths in HTML setup while Andrew and Kyle were experienced enough in SQL to lead the data model creation). Overall, we found that our group worked well together and got along, avoiding some of the troubles that can arise with group members being unmotivated. Despite the clear challenges to working on group projects given current events, group members remained motivated and were willing to work together frequently on the project.

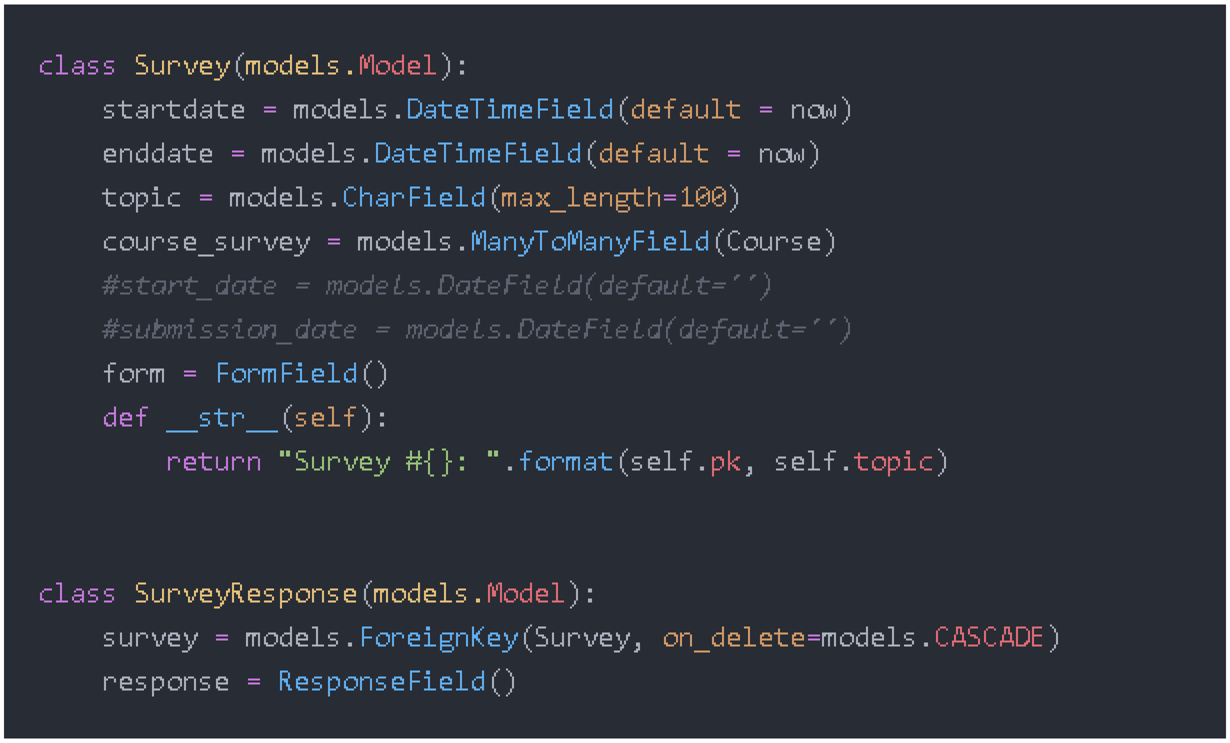
Data Model Outline and Analysis

The system data model focuses on three primary entities: Users (CustomUser), courses (Course), and peer assessments (Survey). Supporting each of these entities are a number of middleman tables that serve to connect the different tables to each other. Each user is defined as either a student or instructor by a boolean field “is_instructor” in the CustomUser model.

These users are created by admins via the Django Administration tools and are joined to courses by the CourseInstructor and CourseStudent tables. Students are also joined to their teams via the CourseTeam table. CourseTeam is linked to its respective course via foreign key while the Course Student table uses ManyToManyFields to aggregate relationships between students and all their respective courses and teams.



```
class CustomUser(AbstractUser):
    username = None
    email = models.EmailField(_('email address'), unique=True)
    first_name = models.CharField(unique = True, max_length = 50)
    last_name = models.CharField(unique = True, max_length = 50)
    eagle_id = models.CharField(max_length = 9, primary_key=True)
    is_instructor = models.BooleanField(default=False)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = []

    objects = CustomUserManager()

    def __str__(self):
        return self.email

class Course(models.Model):
    course_name = models.CharField(max_length = 30)
    course_id = models.CharField(unique = True, max_length = 12, primary_key = True)
    course_date = models.CharField(max_length = 10)
    course_section = models.CharField(max_length = 3)
    course_semester = models.CharField(max_length = 10)
# #
class CourseInstructor(models.Model):
    course = models.ManyToManyField(Course)
    eagle_id = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
# #

# #

class CourseTeam(models.Model):
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    name = models.CharField(unique = True, max_length = 50)

class CourseStudent(models.Model):
    course = models.ManyToManyField(Course)
    eagle_id = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    team = models.ManyToManyField(CourseTeam)
# #
```



```python
class Survey(models.Model):
    startdate = models.DateTimeField(default = now)
    enddate = models.DateTimeField(default = now)
    topic = models.CharField(max_length=100)
    course_survey = models.ManyToManyField(Course)
    #start_date = models.DateField(default='')
    #submission_date = models.DateField(default='')
    form = FormField()
    def __str__(self):
        return "Survey #{}: ".format(self.pk, self.topic)


class SurveyResponse(models.Model):
    survey = models.ForeignKey(Survey, on_delete=models.CASCADE)
    response = ResponseField()
```

This model structure is implemented in the course selection system. Courses are created by (insert entity here), from which point students can enroll in them. Enrolled students generate entities in CourseStudent, from which they can be linked to a team that has been created in the course.

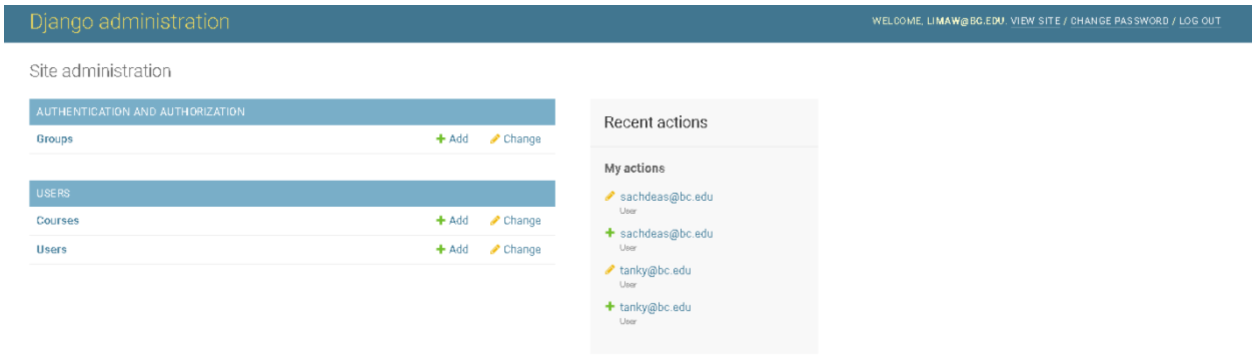
In designing the Survey model, it was determined that while instructors would customize surveys between courses, it was unlikely that they would actually alter survey structure between teams within the same course. As such, some customizability was sacrificed when the Survey model was linked to its respective Course via a ManyToManyField instead of to individual teams, although this change was expected to make implementation of the assessment-course link slightly easier.

In addition to the underlying data model, a number of views were constructed to allow users to actually add information to the data model. As mentioned previously, users were created in Django Administration, but other functions were delegated to their respective users as appropriate. For example, instructors were granted a view that allowed them to create and manage peer assessments. Peer assessments in particular were handled via a prebuilt app called dynamic-django-forms. While using a pre built application would sacrifice some degree of customizability, we felt that the tools it already offered were sufficient for our project and the cost of implementation was not prohibitively high, thus allowing us to save a tangible amount of time.

Requirements Breakdown

Login Requirements - All login requirements pertaining to the main landing page were successfully and fully completed. For ease of use, the login page was built with three different portals, one for each type of user.

Admin Interface - From the admin interface, course creation as well as user administration is handled.



Django administration

WELCOME, LIMAW@BC.EDU. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change

USERS

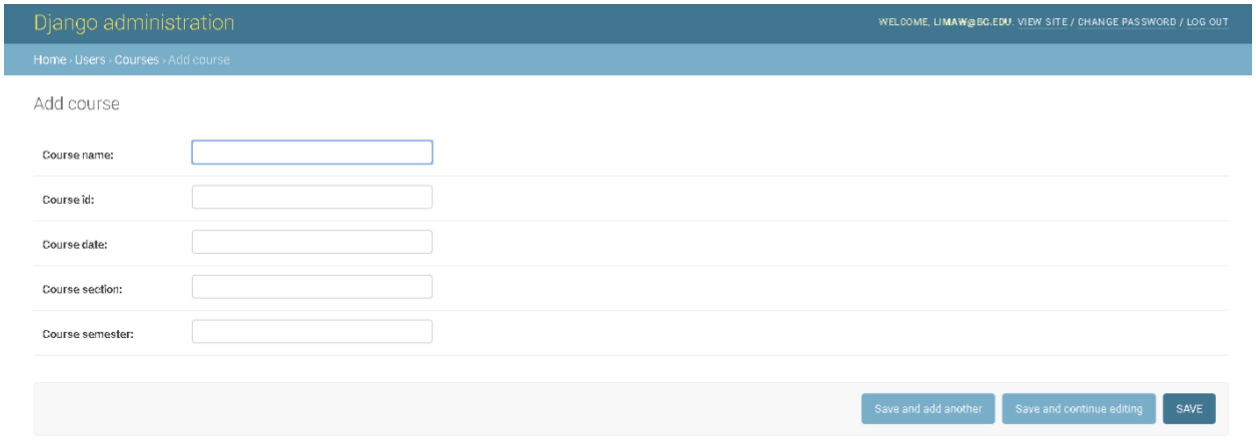
Courses	+ Add	Change
Users	+ Add	Change

Recent actions

My actions

- sachdeas@bc.edu User
- sachdeas@bc.edu User
- tanky@bc.edu User
- tanky@bc.edu User

Course creation interface:



Django administration

WELCOME, LIMAW@BC.EDU. VIEW SITE / CHANGE PASSWORD / LOG OUT

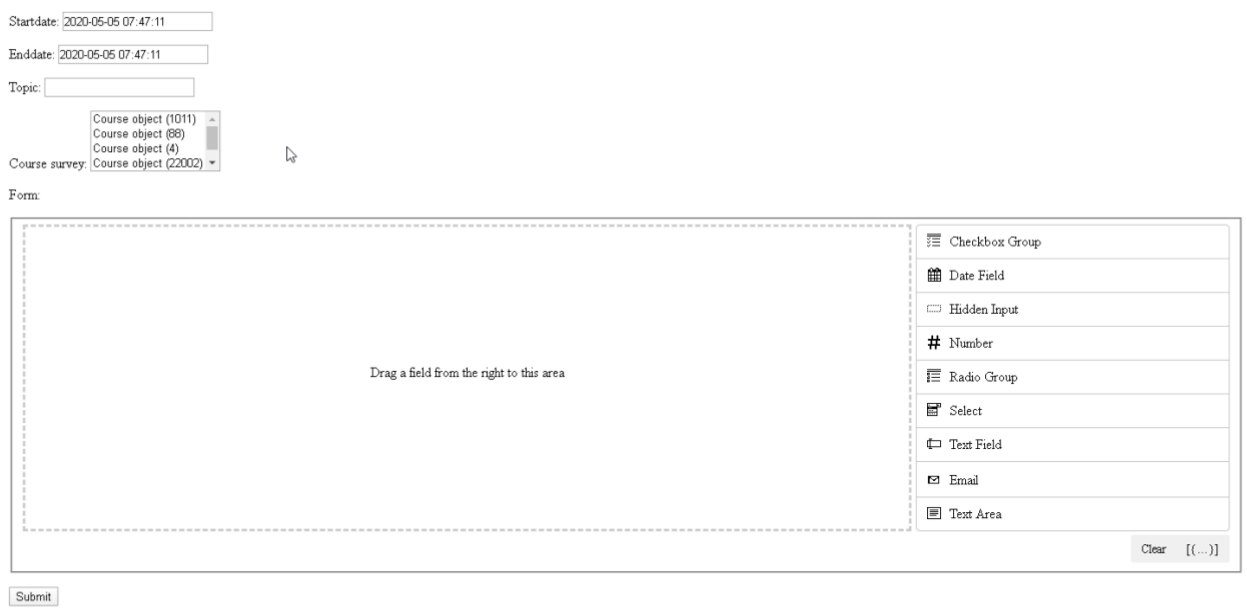
Home › Users › Courses › Add course

Add course

Course name:	
Course id:	
Course date:	
Course section:	
Course semester:	

Save and add another | Save and continue editing | SAVE

Instructor Perspective - Instructors are capable of creating peer assessments with an adequate degree of customizability over their contents. We opted to use the prebuilt app dynamic-django-forms as we felt that it satisfied our requirements without an exceptionally high implementation cost.


Startdate: 2020-05-05 07:47:11

Enddate: 2020-05-05 07:47:11

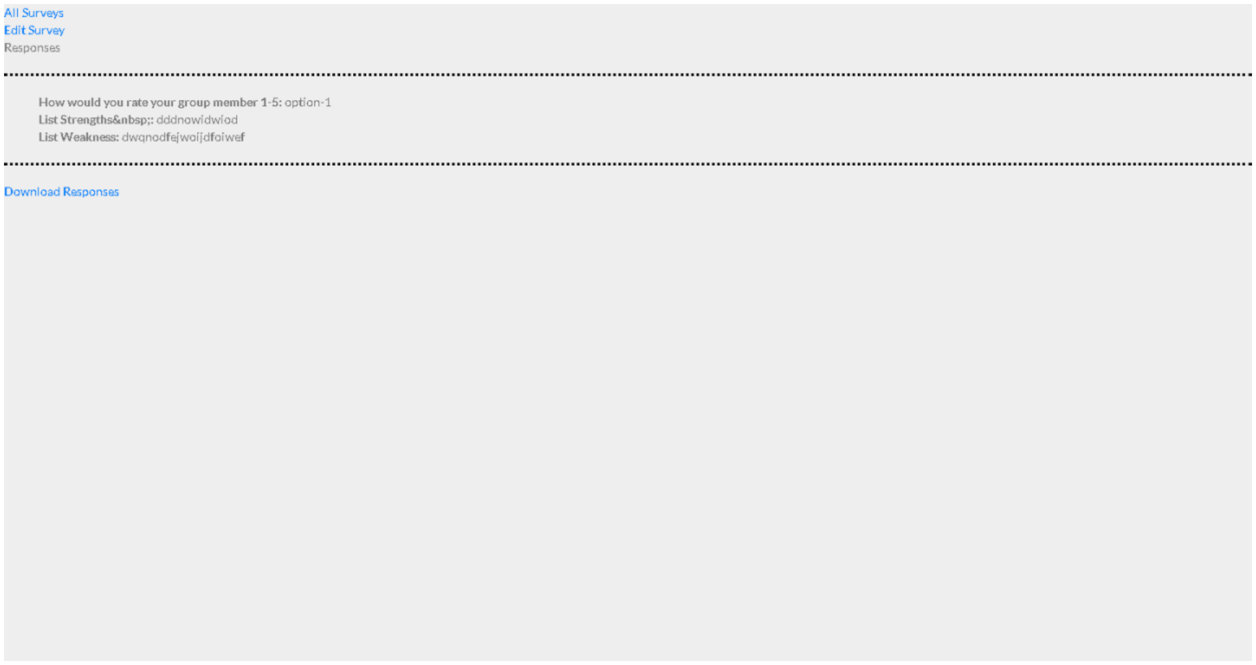
Topic:

Course survey: Course object (1011) Course object (88) Course object (4) Course object (22002)

Form:

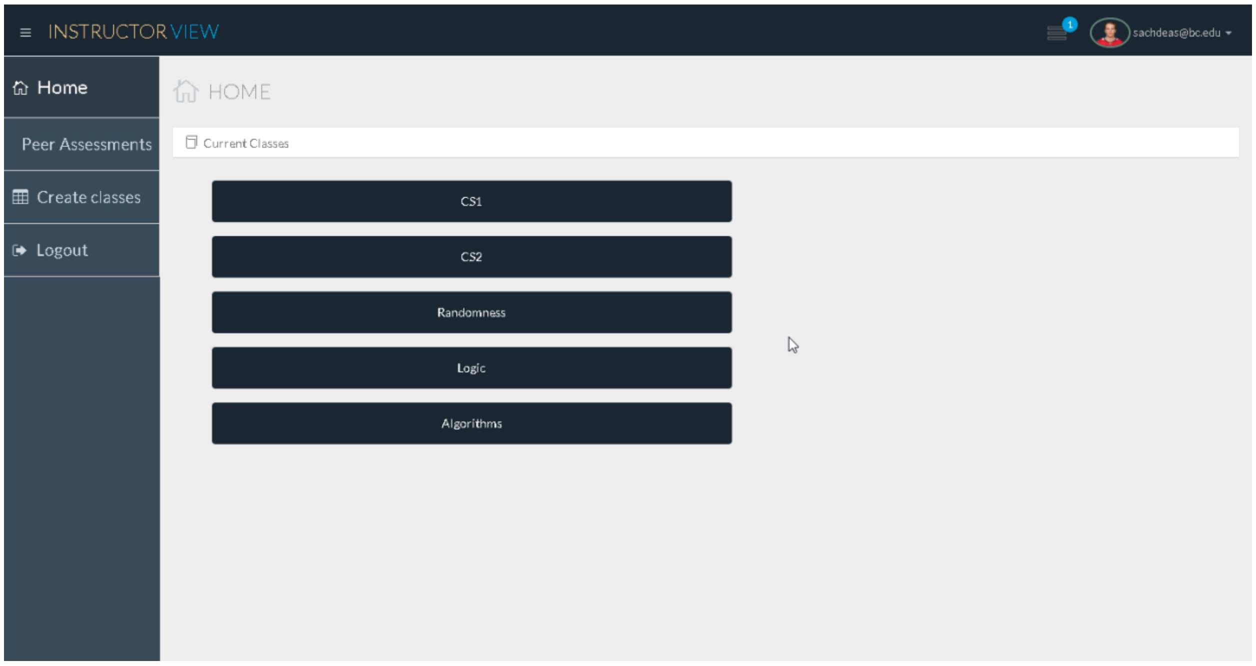
Drag a field from the right to this area

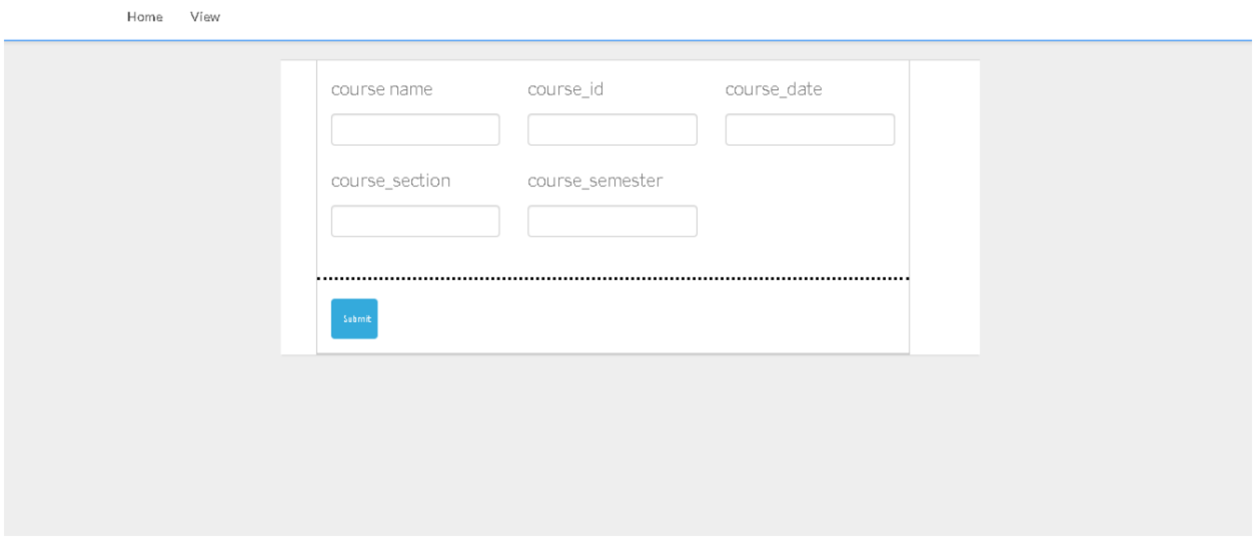
Instructors can view aggregated survey results and download them in an excel format. Excel downloading was handled via the xlwt package.



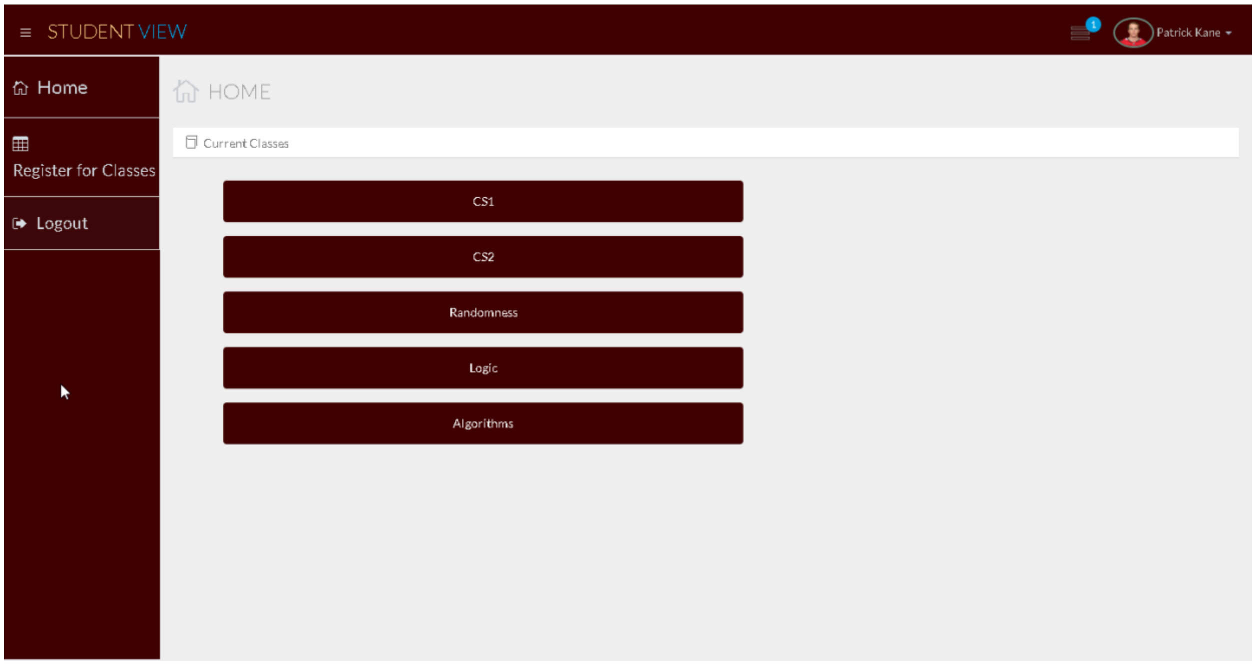
How would you rate your group member 1-5: option-1
List Strengths : dddnowidwiod
List Weakness: dwqnodfejwoijdfoiwef

Instructors can create courses and view them in their landing page as depicted below.





Student Perspective - We were unable to connect students to classes. This caused a chain sequence of requirements being unable to be completed. Without students listed in classes, teams could not be generated from class rosters, and peer assessments couldn't be sent out. As such, a number of requirements related to students were left incomplete.



Generally, we found that multiple aspects of the data model were heavily interlinked and required all other parts to be functioning to work properly. The peer assessment module in particular required user registration and course handling to all be fully functional, in addition to team registration. While we attempted to approach problems from multiple different angles with the objective of completing as many segments as possible, linking these different modules proved difficult.