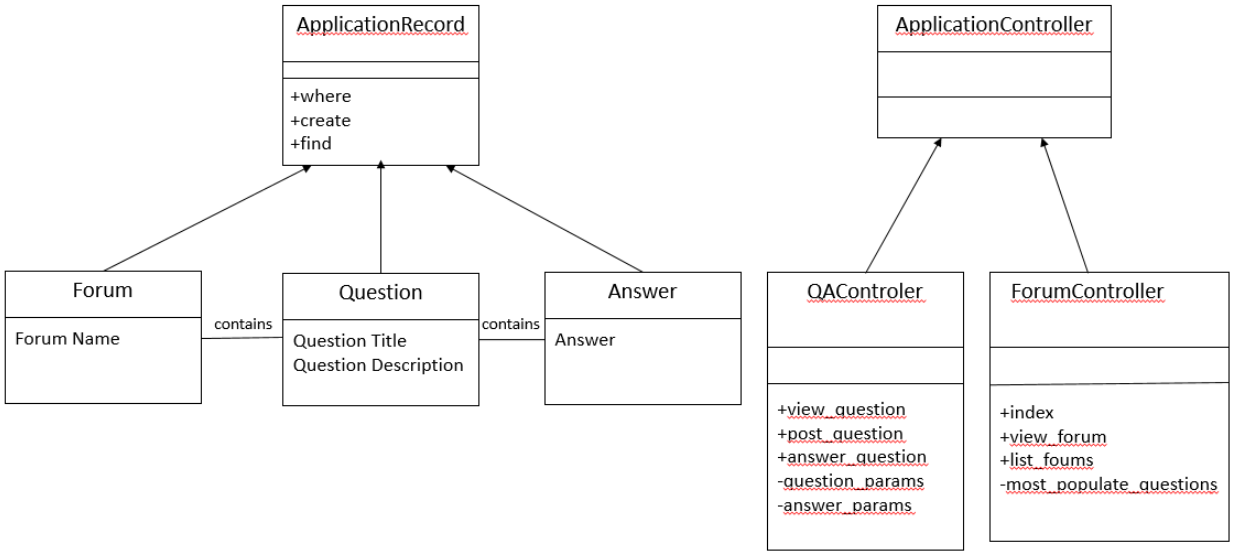
# NKUNet
# Design Document

*Version: Iteration 3*

| Revision History | | |
|---|---|---|
| March 15, 2017 | I5S Team | Initial version |
| March 16, 2017 | I5S Team | Revise ERD |
| April 2, 2017 | I5S Team | Iteration 2 |
| April 3, 2017 | I5S Team | Minor updates, include Sequence Diagrams for UC5, UC6, UC11 and UC12. |
| April 10, 2017 | I5S Team | Added Iteration 2 database commands, updated ERD |
| May 1, 2017 | IS5 Team | Added Sequence Diagrams for remaining use cases, exit criteria for Iteration 3 and section on GoF design patterns. |

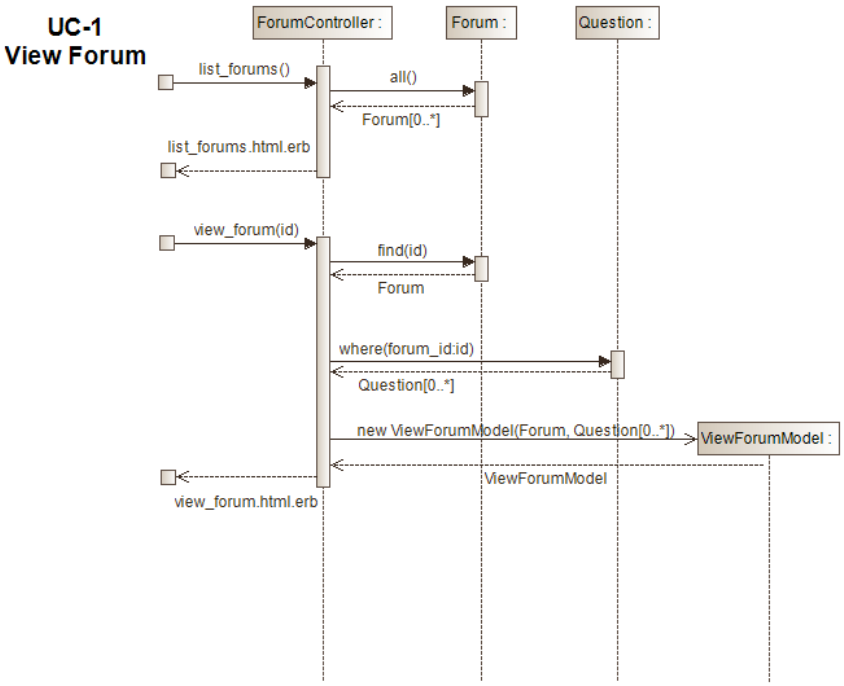
## Table of Contents

# Class Diagram

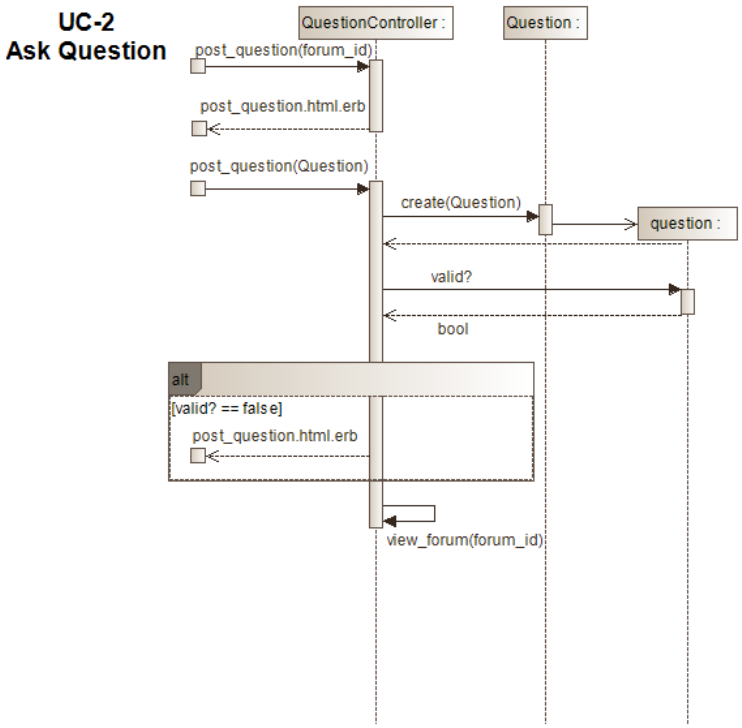



# Sequence Diagrams

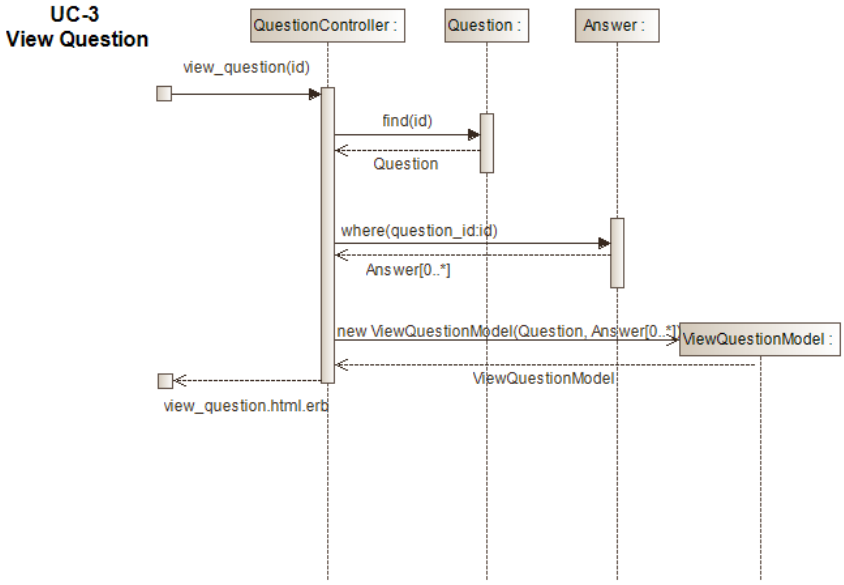
## UC-1



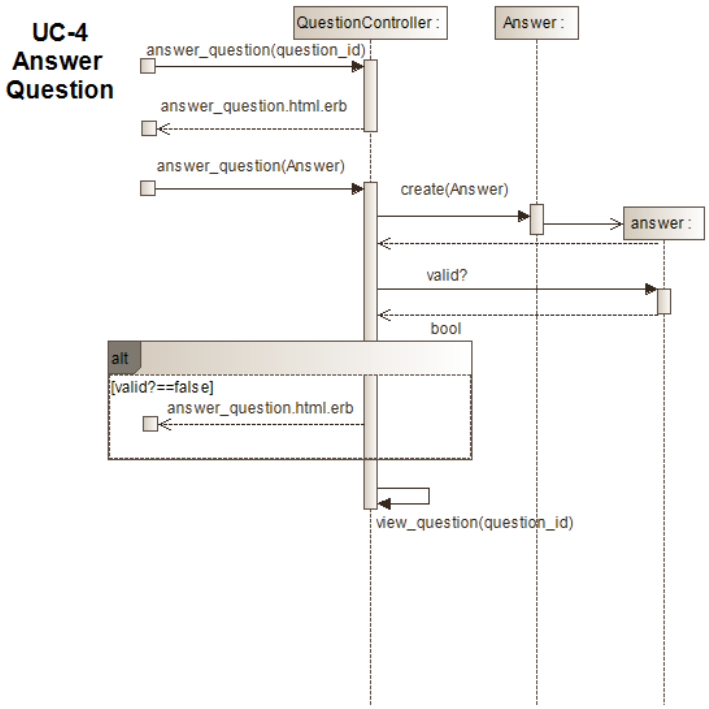
## UC-2


UC-2
Ask Question
QuestionController :
Question :
post_question(forum_id)
post_question.html.erb
post_question(Question)
create(Question)
question :
valid?
bool
alt
[valid? == false]
post_question.html.erb
view_forum(forum_id)

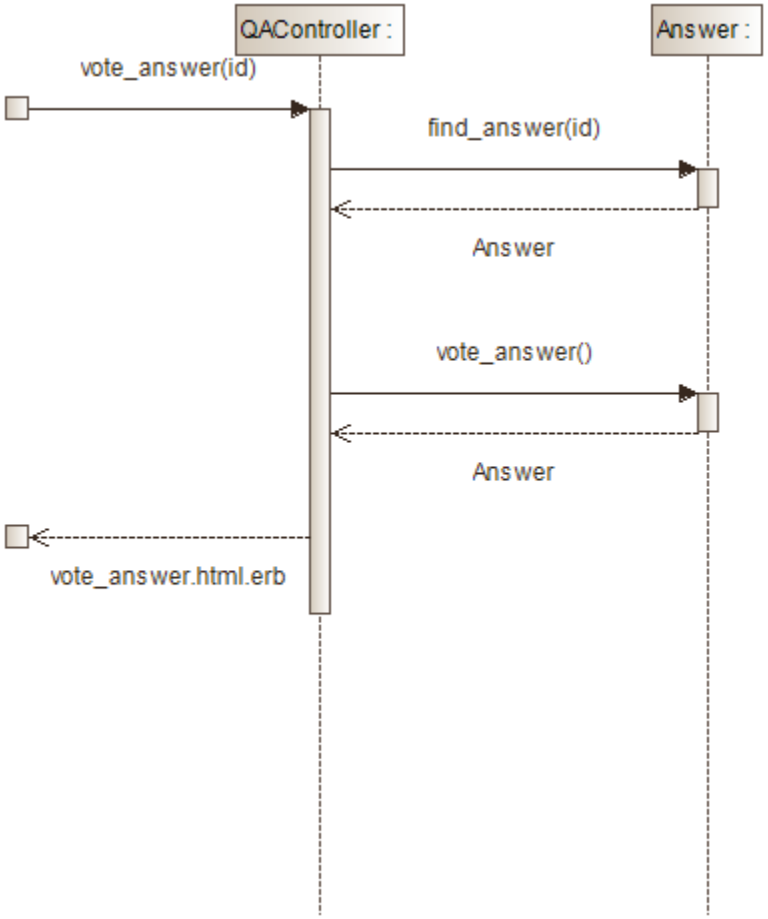
## UC-3


UC-3
View Question
QuestionController :
Question :
Answer :
view_question(id)
find(id)
Question
where(question_id:id)
Answer[0..*]
new ViewQuestionModel(Question, Answer[0..*])
ViewQuestionModel :
ViewQuestionModel
view_question.html.erb

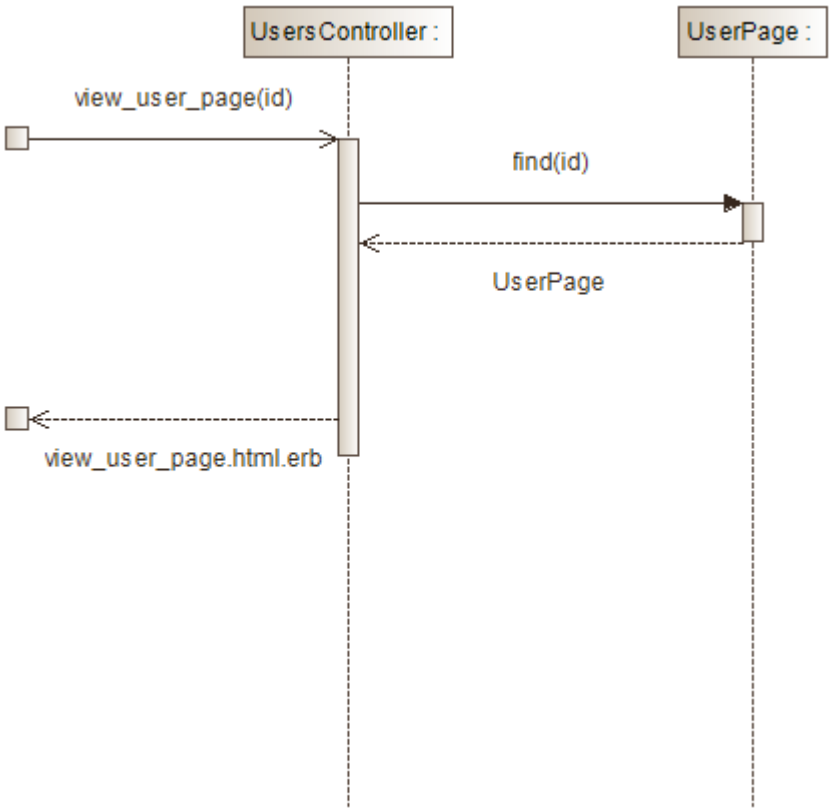
## UC-4


UC-4
Answer
Question
QuestionController :
Answer :
answer_question(question_id)
answer_question.html.erb
answer_question(Answer)
create(Answer)
answer :
valid?
bool
alt
[valid?==false]
answer_question.html.erb
view_question(question_id)

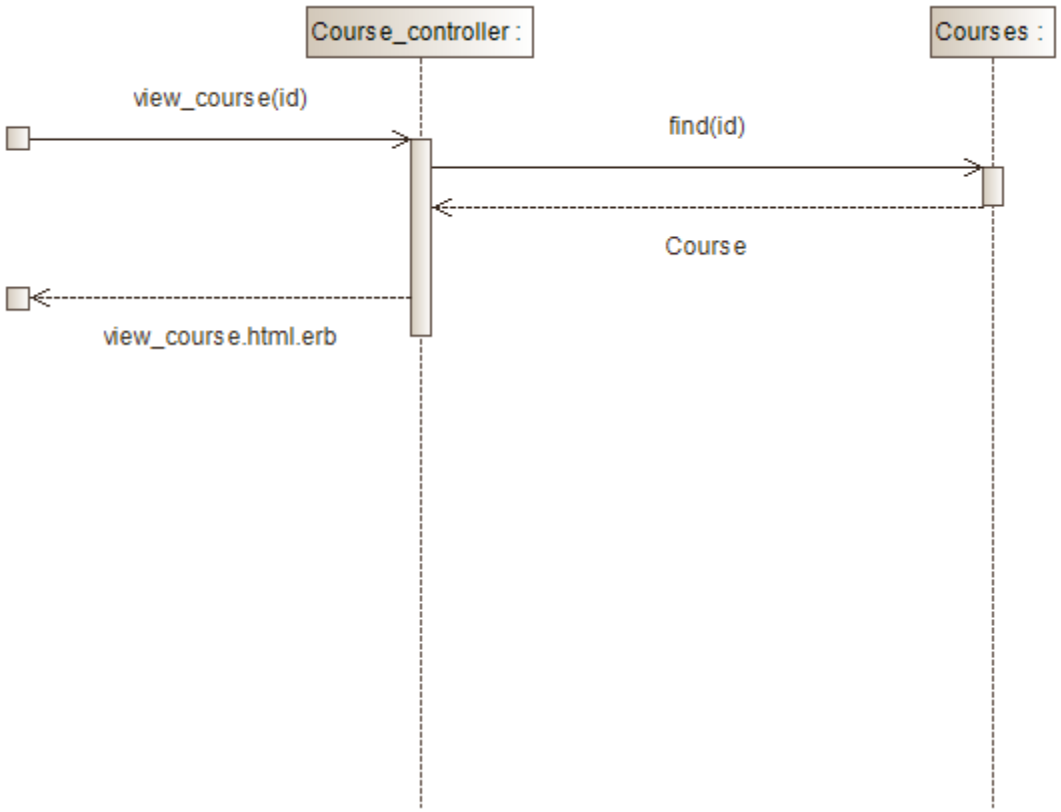
## UC-5


QAController :
Answer :
vote_answer(id)
find_answer(id)
Answer
vote_answer()
Answer
vote_answer.html.erb

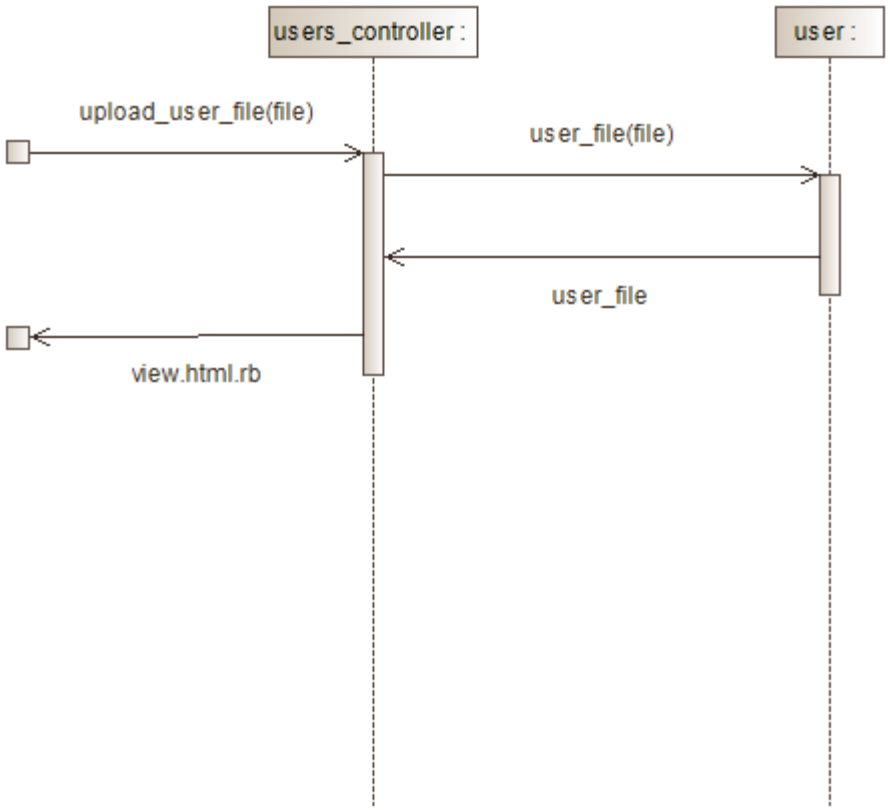
## UC-6


UsersController :
UserPage :
view_user_page(id)
find(id)
UserPage
view_user_page.html.erb

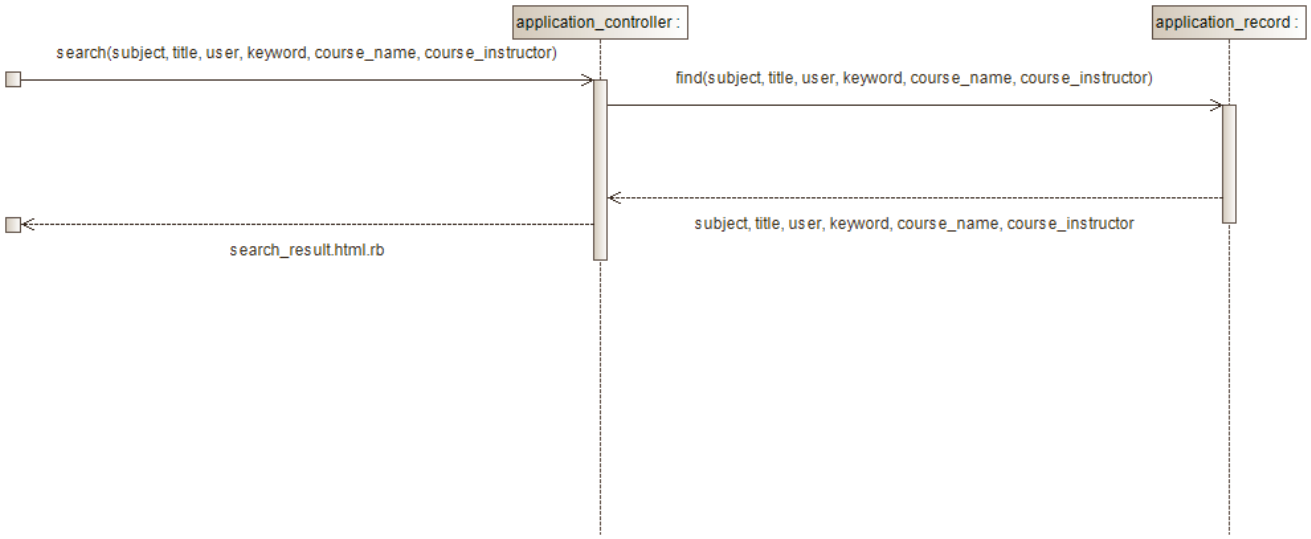
## UC-7


Course_controller :
Courses :
view_course(id)
find(id)
Course
view_course.html.erb

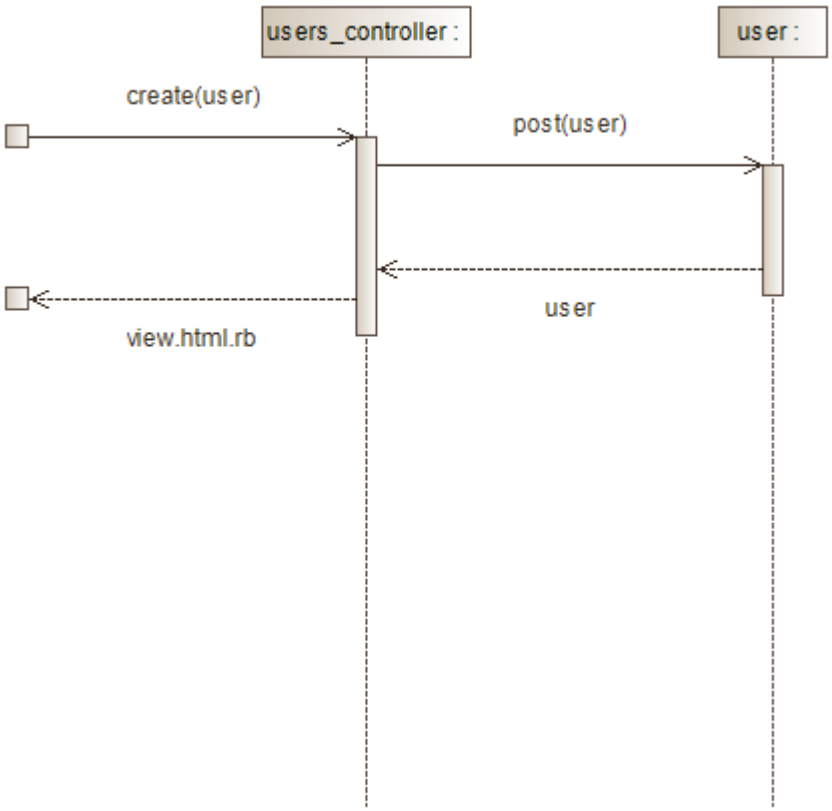
## UC-8


users_controller :
user :
upload_user_file(file)
user_file(file)
user_file
view.html.rb

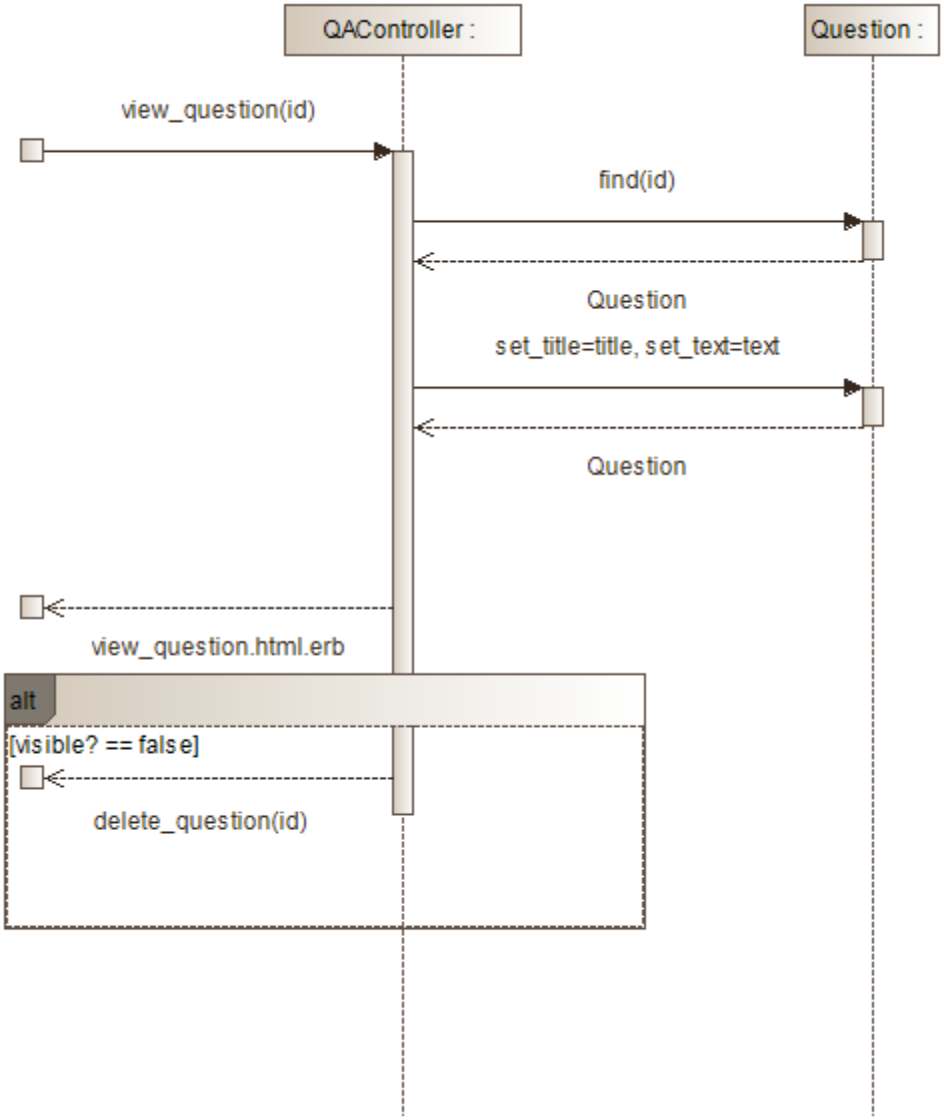
## UC-9


application_controller :
application_record :
search(subject, title, user, keyword, course_name, course_instructor)
find(subject, title, user, keyword, course_name, course_instructor)
subject, title, user, keyword, course_name, course_instructor
search_result.html.rb

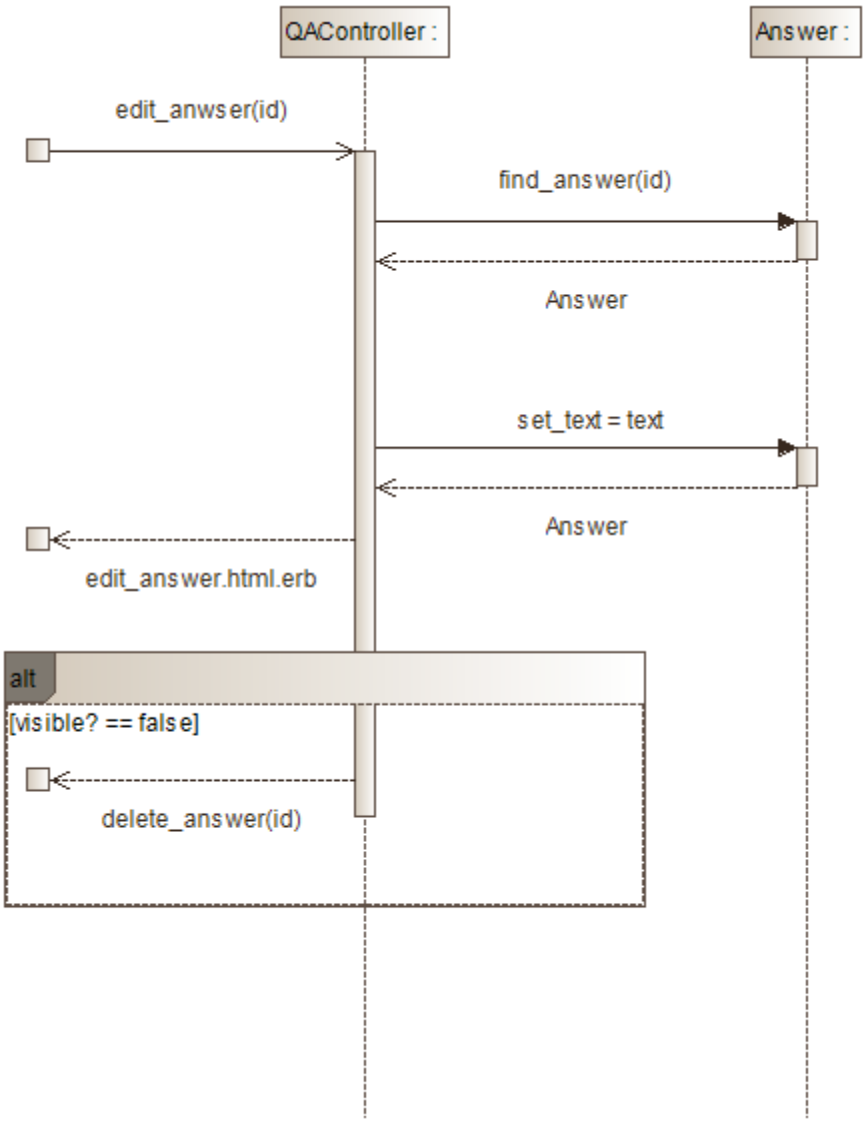
## UC-10


users_controller :
user :
create(user)
post(user)
user
view.html.rb

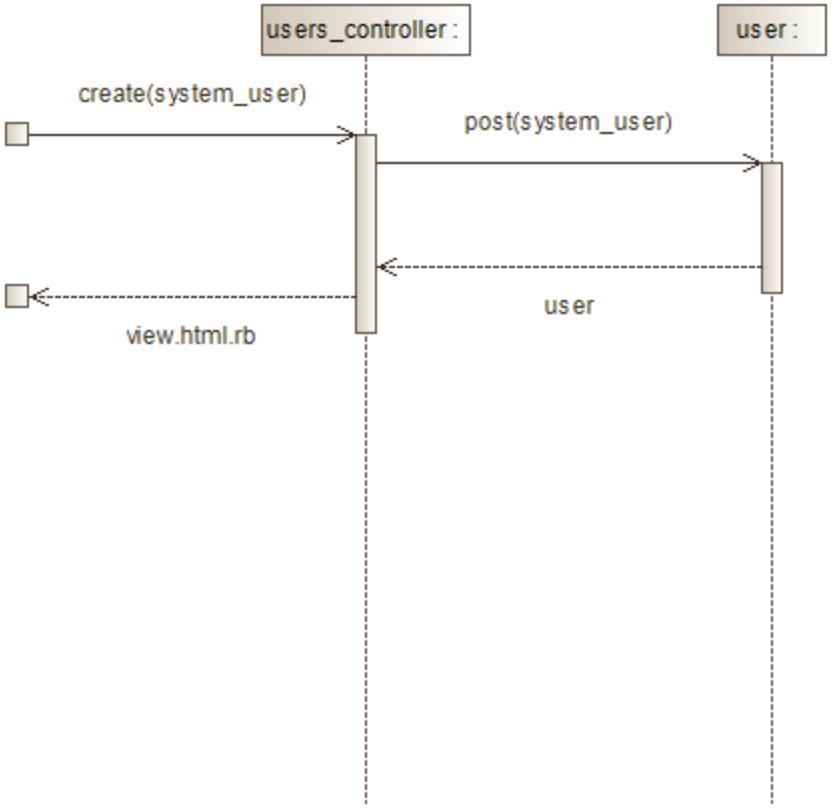
## UC-11


QAController :
Question :
view_question(id)
find(id)
Question
set_title=title, set_text=text
Question
view_question.html.erb
alt
[visible? == false]
delete_question(id)

## UC-12

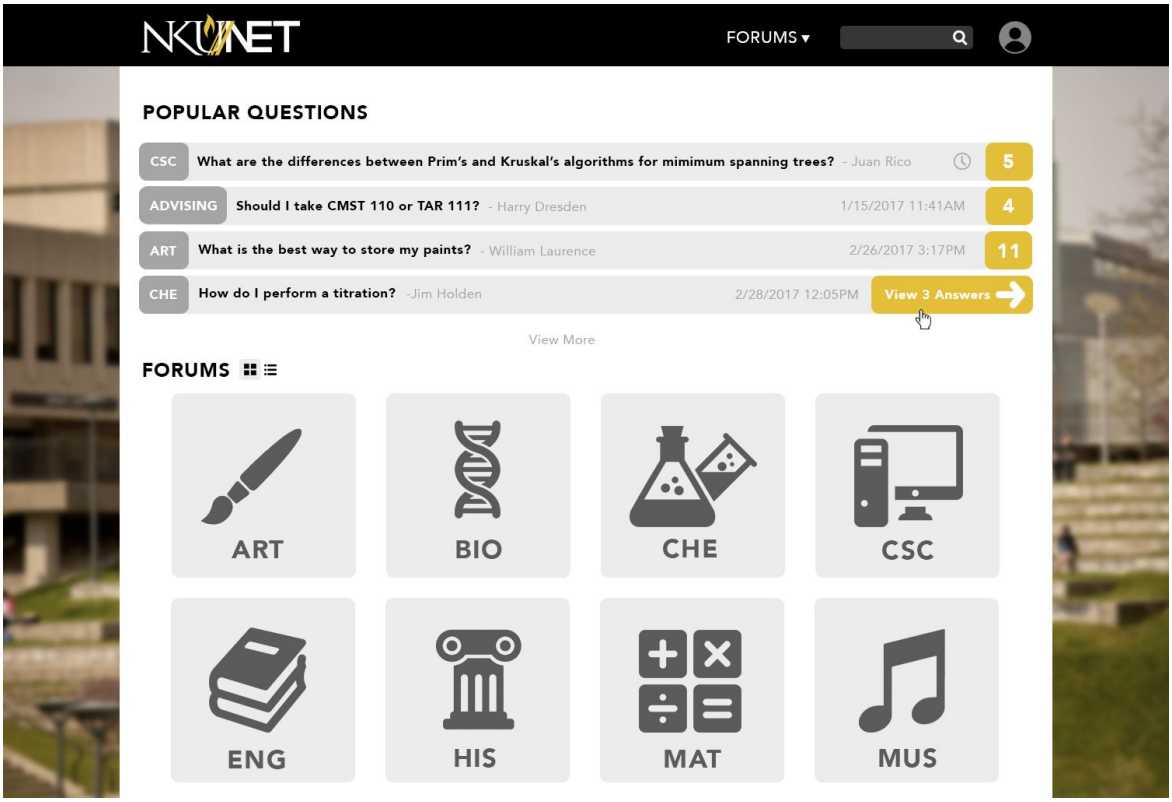

QAController :
Answer :
edit_anwser(id)
find_answer(id)
Answer
set_text = text
Answer
edit_answer.html.erb
alt
[visible? == false]
delete_answer(id)

## UC-13


users_controller :
user :
create(system_user)
post(system_user)
user
view.html.rb

# Design Decisions

## UI Design




## System Design

### Iteration 1

In Iteration 1, we focused mainly on learning how to implement the Rails framework. We chose use cases that were essential, but relatively straight-forward with limited alternate paths. Other factors driving our design decisions were the availability of resources and the deadline.

The use cases selected were:

- View Forum
- Ask Question
- View Question
- Answer Question

It was our desire to create an aesthetically pleasing website that is easy to navigate, looks Nuevo and has familiar functionality to many of today's websites. However, as we progressed through Iteration 1 we realized we needed to start looking at the functionality before the aesthetics. It is our hope to be able to incorporate a more interactive UI in a later iteration.

No one on our team had experience with either Ruby or Rails. The learning curve proved to be steeper than we had anticipated. To get us all on the same footing, we decided to group code with all team members present. While this proved to be very productive, it was a time drain on our development cycle.

One item of interest is that the object we have been calling Resource has been implemented as UserFile. Apparently, "resource" is a reserve word and could not be used as a database object.

### Iteration 2

In Iteration 2, one of our main focuses was on user authentication. We wanted to have a reasonable level of security built into our system and so decided to implement the authlogic plugin.

The use cases selected for this iteration were:

- Vote on Answer
- View User Page
- Edit Question
- Edit Answer

Group coding proved to be more productive for our team in this iteration.

### Iteration 3

In Iteration 3, we completed the remaining use cases.

## GRASP Patterns

The tables we have created make use of the following GRASP patterns. The following is presented from a design perspective only.

### Creator

The following classes play the role of creator:

- User – responsible for creating User_File
- Forum – responsible for creating Question
- Question – responsible for creating Answer, Question_Keyword and Keyword
- Answer – responsible for creating Vote
- Section – responsible for creating Student_Section

### Controller

The following classes play the role of controller:

- User – performs user authentication and controls what functionality is available based on Role
- Section – takes in a file of student registration and creates Student_Section instances

### Information Expert

The following classes play the role of information expert:

- User – makes use of Question and Vote to present a user's reputation

### Low Coupling

All classes in our design were created with low coupling in mind.

### High Cohesion

All classes in our design were created with high cohesion in mind.

### Polymorphism

The following classes demonstrate polymorphic behavior:

- User – will render User Page differently based on Role. Only Students and Faculty will have the capability of adding Resources.

### Indirection

The following classes demonstrate indirection:

- Question_Keyword – transition table between Question and Keyword that reduces a many-to-many relationship
- Student_Section – transition table between User and Section that reduces a many-to-many relationship

### Pure Fabrication

The following classes were fabricated to reduce coupling and raise cohesion:

- Question_Keyword
- Student_Section

### Protected Variations

The following classes support protected variations:

- Role – allows User to provide the appropriate capabilities to different types of users

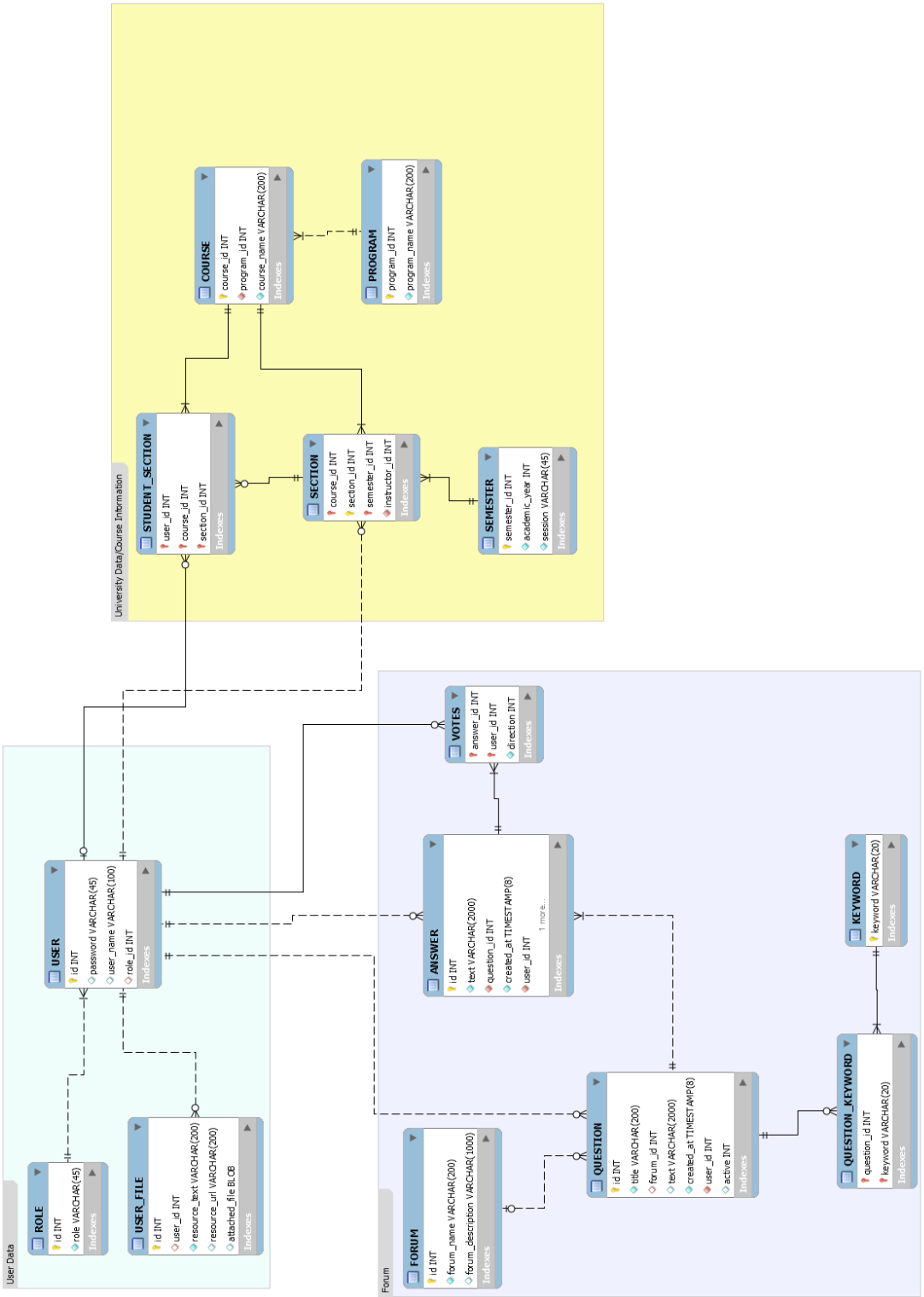
## GoF Design Patterns

We designed our database tables fully during Iteration 1. As this was before we discussed any of the GoF patterns, we did not go back and redesign in an attempt to include any of the GoF design patterns.

In actuality, the database for this project is relatively straight-forward and we do not see where there would be opportunities to include GoF patterns in our database design. On the other hand, the Rails framework is resplendent with examples of various GoF patterns. Some that we have recognized are Façade, Composite, Singleton, Adapter and Iterator.

# Database Design

## ERD



## Database Discussion

Our database model falls into roughly three parts:
1. User-related tables (represented in green section of ERD)
2. Course-related tables (represented in yellow of ERD)
3. Forum-related tables (represented in purple ERD)

All tables are in third-normal form.

### Iteration 1

Iteration 1 focused mainly on Forum-related tables. In this iteration, we implemented Forum, Question and Answer. We also implemented User, Role and Resource as some of the other tables have foreign keys into the User table. It was advantageous to define these tables now, even though we will not be implementing their functionality until a later iteration.

### Iteration 2

Iteration 2 builds on the user functionality and includes authentication. The role of Administrator is also introduced with functionality to edit/delete both questions and answers. In this iteration, we implemented the Vote class as well as the function of up-voting / down-voting an answer.

### Iteration 3

The only thing changed in Iteration 3 as relates to the database model is the addition of a section_name field in the Section table.

## Rails Database Commands

### Iteration 1

- `rails generate model Role role:string`
- `rails generate model User user_name:string password:string role:references`
- `rails generate model UserFile resource_text:string resource_url:string attached_file:binary user:references`
- `rails generate model Forum forum_name:string forum_description:string resource_url:string`
- `rails generate model Question title:string text:string active:boolean user:references forum:references  #change migration script::> t.boolean :active, :default => true`
- `rails generate model Answer text:string active:boolean user:references question:references  #change migration script::> t.boolean :active, :default => true`
- `rake db:reset`

### Iteration 2

- `rails generate model Role role:string`
- `rails generate model User user_name:string password:string role:references #change migration script::> t.intger :role_id, :default => 2`
- `rails generate model UserFile resource_text:string resource_url:string attached_file:binary user:references`
- `rails generate model Forum forum_name:string forum_description:string resource_url:string`

- `rails generate model Question title:string text:string active:boolean user:references forum:references  #change migration script::> t.boolean :active, :default => true`
- `rails generate model Answer text:string active:boolean user:references question:references  #change migration script::> t.boolean :active, :default => true`
- `rails generate model Votes direction:integer answers:references user:references #modify database w/`
- `rails generate model Keyword keyword:string`
- `rails generate model QuestionKeyword question:references keyword:references`
- `rails generate model Semester academic_year:integer session:string`
- `rails generate model Section semester:references course:references user:references`
- `rails generate model Program program_name:string`
- `rails generate model Course course_name:string program:references`
- `rails generate model StudentSection user:references section:references course:references`

### Iteration 3

- `rails generate model Role role:string`
- `rails generate model User user_name:string password:string role:references #change migration script::> t.intger :role_id, :default => 2`
- `rails generate model UserFile resource_text:string resource_url:string attached_file:binary user:references`
- `rails generate model Forum forum_name:string forum_description:string resource_url:string`
- `rails generate model Question title:string text:string active:boolean user:references forum:references  #change migration script::> t.boolean :active, :default => true`
- `rails generate model Answer text:string active:boolean user:references question:references  #change migration script::> t.boolean :active, :default => true`
- `rails generate model Votes direction:integer answers:references user:references #modify database w/`
- `rails generate model Keyword keyword:string`
- `rails generate model QuestionKeyword question:references keyword:references`
- `rails generate model Semester academic_year:integer session:string`
- `rails generate model Section section_name:string semester:references course:references user:references`
- `rails generate model Program program_name:string`
- `rails generate model Course course_name:string program:references`
- `rails generate model StudentSection user:references section:references course:references`

## Exit Criteria

1. System must compile without errors.
2. 95% of the following test cases must pass.
3. Test Cases for Iteration 1.
    a. The Home Page displays the latest Questions that have been asked and a list of Forums.

   b. A specific Forum is displayed when selected from the Home Page. The Forum will contain a list of Questions that have been posted.
   c. A new Question can be entered into a Forum by supplying the Question Title and Question Description. The Forum will be refreshed automatically so that the Question is visible.
   d. Attempting to add a Question without supplying a Question Title will leave you in enter mode for that Question.
   e. Attempting to add a Question without supplying a Question Description will leave you in enter mode for that Question.
   f. Selecting a Question will display the list of Answers for it.
   g. A new Answer can be entered into a Question by supplying the answer text. The Question will be refreshed automatically so that the Answer is visible.
   h. Attempting to add an Answer without supplying an Answer Text will leave you in enter mode for that Answer.
4. Test Cases for Iteration 2.
   a. Up-vote on an answer is recorded. Display in forum and on User Page are updated.
   b. Down-vote on an answer is recorded. Display in forum and on User Page are updated.
   c. If a user votes on an answer multiple times, only the last vote is kept.
   d. User Page is viewable. A User Page contains: classes where student is/was enrolled, reputation (number of questions posted, number of up-votes, number of down-votes), most recent questions posted, most recent answers posted and all resources posted.
   e. User Page is reached by clicking User link on a Question.
   f. User Page is reached by clicking User link on an Answer.
   g. Question is editable by an Administrator.
   h. Question is not editable if not an Administrator.
   i. Question can be deleted by Administrator.
   j. Question cannot be deleted if not an Administrator.
   k. When question is deleted, all answers under that question are also deleted.
   l. When question is deleted, voting for all answers under that question are also deleted.
   m. Answer is editable by an Administrator.
   n. Answer is not editable if not an Administrator.
   o. Answer can be deleted by Administrator.
   p. Answer cannot be deleted if not an Administrator.
   q. When answer is deleted, voting for that answer is also deleted.
5. Test Cases for Iteration 3.
   a. Class Page is viewable. A Class Page contains a list of sections.
   b. Class Section Page is viewable. A Class Section Page contains: instructor and a list of registered students. Each of these is a link that the faculty/student's page.
   c. Class Page is reached by clicking Class link on a User Page.
   d. Class Section Page is reached by clicking Section link on a Class Page.
   e. User Page now include Resources.
   f. User can select Resource Upload to add a resource. Resources have a title and a blob area, which will hold a smallish file.
   g. User does not have option of Resource Upload when viewing someone else's User Page.

h. Other users can view resource information when viewing User Page.
i. Search page is viewable. Search allows Users to find courses or questions.
j. Search can find courses by course name. A list of course sections is returned.
k. Search can find courses by instructor. A list of course sections is returned.
l. Search can find courses by a combination of course name and instructor.
m. Search can find questions by subject. A list of questions is returned.
n. Search can find questions by user. A list of questions is returned.
o. Search can find questions by post title. A list of questions is returned.
p. Search can find questions by keyword. A list of questions is returned.
q. Search can find questions by a combination of subject, user, post title and keyword.
r. Registrar can upload a file to load course information. The following tables may be updated:
    i. Semester (if not already created)
    ii. Course (if a new course)
    iii. Section (if a new section)
    iv. User and Student_Section for students (Student name, email address, default password)
    v. User and Section for faculty (Faculty name, email address, default password)
s. Administrator can add an Administrator.
t. Administrator can add a Registrar.
u. A User (Student, Faculty) cannot add an Administrator.
v. A User (Student, Faculty) cannot add a Registrar.
w. A Registrar cannot add a User (Student, Faculty).
x. A Registrar cannot add an Administrator.
y. A Registrar cannot add a Registrar.

6. Approval from Mr. McCord based on our presentation.
7. General approval of other project teams based on our presentation.