Team Super Ghost

Iteration 2

Hudson, Nathaniel - Jansen, Shane - Owens, Kirby - Zumwalde, Zach

Vision

(No new updates to vision)

Requirements

Requirements

Use Cases Chosen:

- (UC6) Edit Question
- (UC7) Upvote/Downvote Answer
- (UC8) Upvote/Downvote Question
- (UC9) Authenticate User

UC6: Edit Question

Scope: Edit Question System
Level: user-goal
Primary Actor: Administrator
Stakeholders and Interests:

- Administrator: Wants to be able to edit question on NKU-Net.
- School: Wants questions to be appropriate and helpful to students.

Preconditions: Administrator is logged into system upon being identified and authenticated with admin permissions.
Success guarantee (or Postconditions): Administrator's updated edit of the question is uploaded to NKU-Net.

Main Success Scenario (or Basic Flow):

1. Administrator selects a class forum.
2. Administrator selects a question that he/she wants to edit.
3. Administrator edits question to his/her liking.
4. Administrator saves question.

Extensions (or Alternative Flows):
*a. At any time, System fails:

1. To ensure security of system, Administrator is logged off and connection is shut off.

Special Requirements:

- System posts edited question within 30 seconds, 90% of the time.
- Compatible with all major Internet browsers.

Technology and Data Variations List:
*a. Administrator can use any major web browser (Firefox, Google Chrome,
Safari, Microsoft Edge, etc.).

Frequency of Occurrence: Could be nearly continuous.

UC7: Upvote/Downvote Answer

Scope: Upvote/Downvote Question System
Level: user-goal
Primary Actor: Student, Faculty Member
Stakeholders and Interests:

- Student: Wants to receive credit/incentive for question.
- Faculty: Wants to encourage students into posting good questions.
- Administrator: Wants good posting techniques to be encouraged.
- School: Wants good posting techniquest to be encouraged.

Preconditions: Student/Faculty Member is logged into system upon being identified and authenticated.
Success guarantee (or Postconditions): Question's upvote/downvote information is updated in NKU-Net.
Main Success Scenario (or Basic Flow):

1. Student/Faculty Member selects a class forum.
2. Student/Faculty Member selects a question in forum.
3. Student/Faculty Member upvotes or downvotes a question.
4. System updates the vote count for the question.

Extensions (or Alternative Flows):
*a. At any time, System fails:

1. To ensure security of system, Administrator is logged off and connection is shut off.

Special Requirements:

- System upvotes/downvotes question within 30 seconds, 90% of the time.
- Compatible with all major Internet browsers.

Technology and Data Variations List:
*a. Administrator can use any major web browser (Firefox, Google Chrome,
Safari, Microsoft Edge, etc.).

Frequency of Occurrence: Could be nearly continuous.

UC8: Upvote/Downvote Question

Scope: Upvote/Downvote Answer System
Level: user-goal
Primary Actor: Student/Faculty Member
Stakeholders and Interests:

- Student: Wants to be able to receive credit/incentive for answer.
- Faculty Member: Wants to encourage students into posting good answers.
- Administrator: Wants to encourage students into posting good answers.
- School: Wants answers that are the most helpful to be seen first.

Preconditions: Student/Faculty Member is logged into system upon being identified and authenticated.
Success guarantee (or Postconditions): Answer's upvote/downvote information is updated on NKU-Net.
Main Success Scenario (or Basic Flow):

1. Student/Faculty Member selects a class forum.
2. Student/Faculty Member selects a question forum.
3. Student/Faculty Member upvotes/downvotes the answer.
4. System updates the vote count for the answer.

Extensions (or Alternative Flows):
*a. At any time, System fails:

1. To ensure security of system, Administrator is logged off and connection is shut off.

Special Requirements:

- System updates votes within 30 seconds, 90% of the time.
- Compatible with all major Internet browsers.

Technology and Data Variations List:
*a. Administrator can use any major web browser (Firefox, Google Chrome,
Safari, Microsoft Edge, etc.).

Frequency of Occurrence: Could be nearly continuous.

UC9: Authenticate User

Scope: Authenticate System
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- <u>Students</u>: Wants to login as fast as possible.
- <u>Faculty</u>: Wants to login as fast as possible.
- <u>Administrator</u>: Wants login to be secure.
- <u>School</u>: Wants students to not be able to modify each other's data.

Preconditions: User already has an account created.
Success guarantee (or Postconditions): Administrator's updated edit of the answer is uploaded to NKU-Net.

Main Success Scenario (or Basic Flow):

1. User types in email address and password into login page.
2. System authenticates user.
3. System notifies user of successful login and redirects them to course page.

Extensions (or Alternative Flows):
*a. At any time, System fails:

1. To ensure security of system, Administrator is logged off and connection is shut off.

2a. Authentication Fails

1. System notifies user of authentication failure.

Special Requirements:

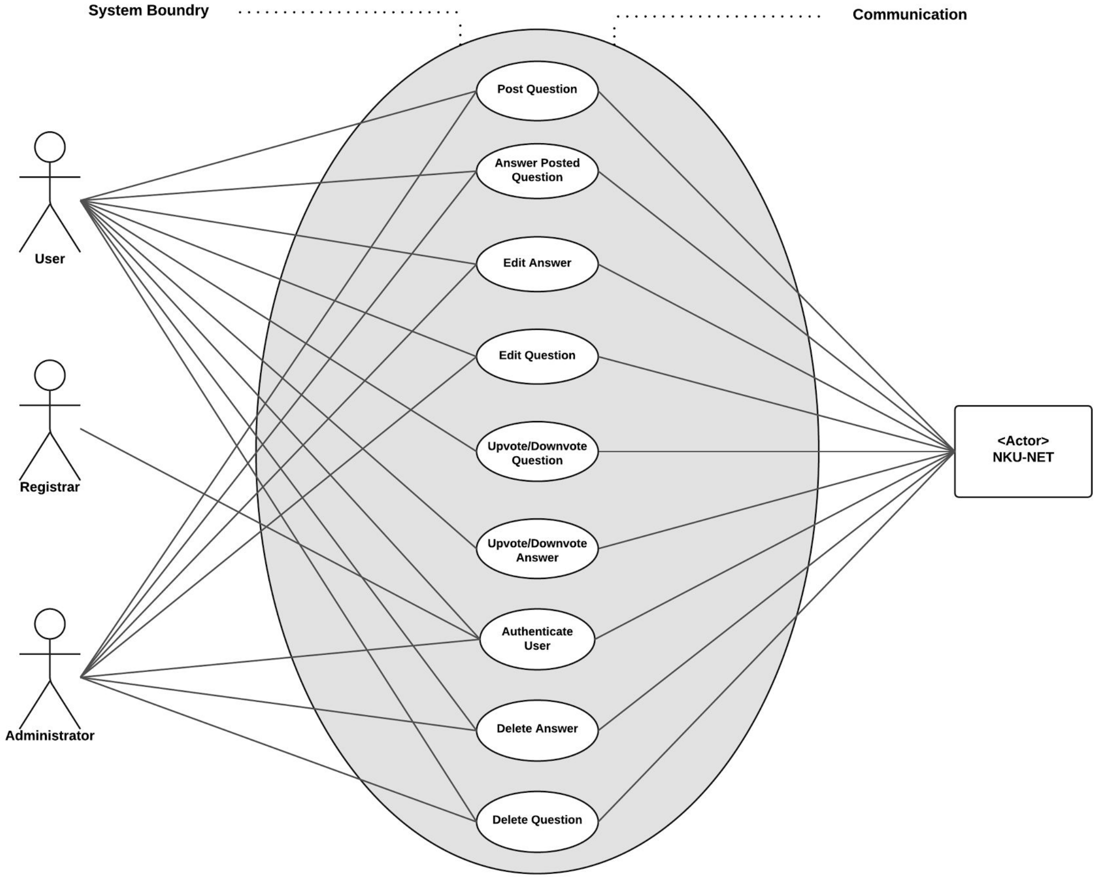
- System posts edited answer within 30 seconds, 90% of the time.
- Compatible with all major Internet browsers.

Technology and Data Variations List:
*a. Administrator can use any major web browser (Firefox, Google Chrome,
Safari, Microsoft Edge, etc.).

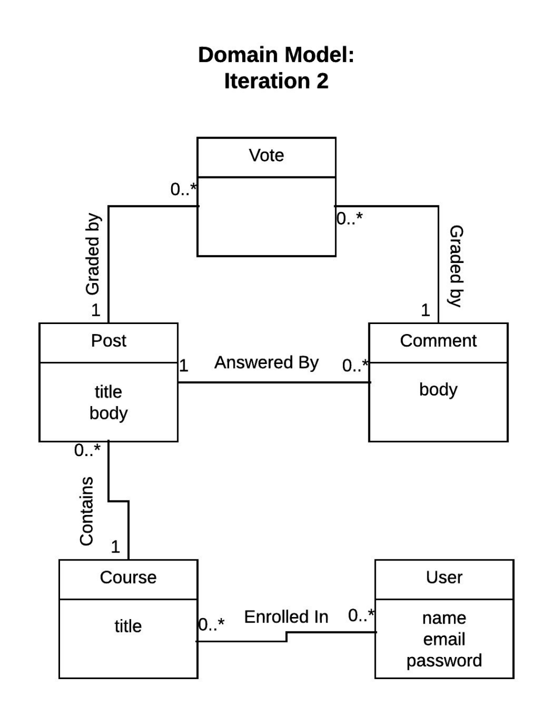
Frequency of Occurrence: Could be nearly continuous.

Use Case Model

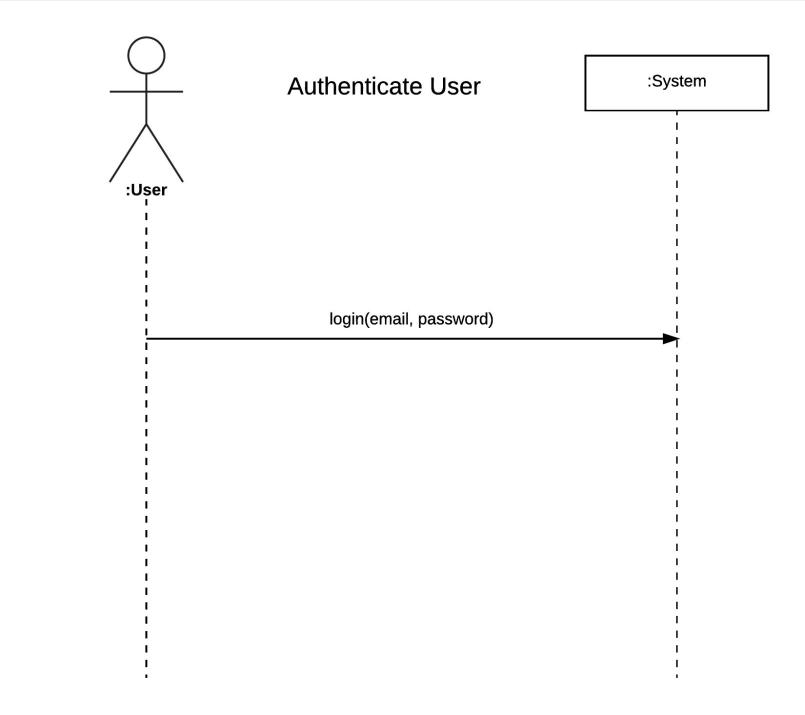
USE CASE DIAGRAM



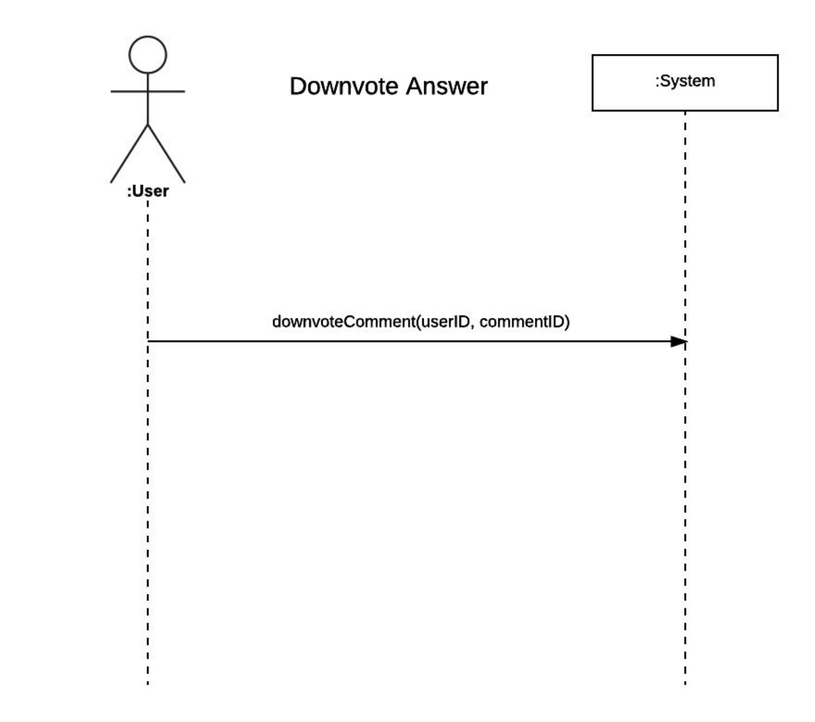
Domain Model



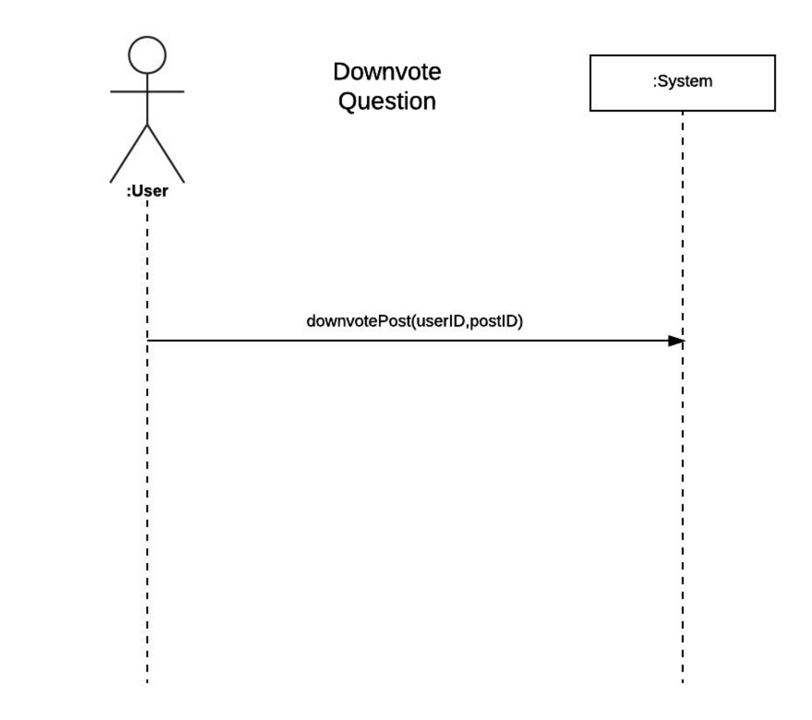
System Sequence Diagrams



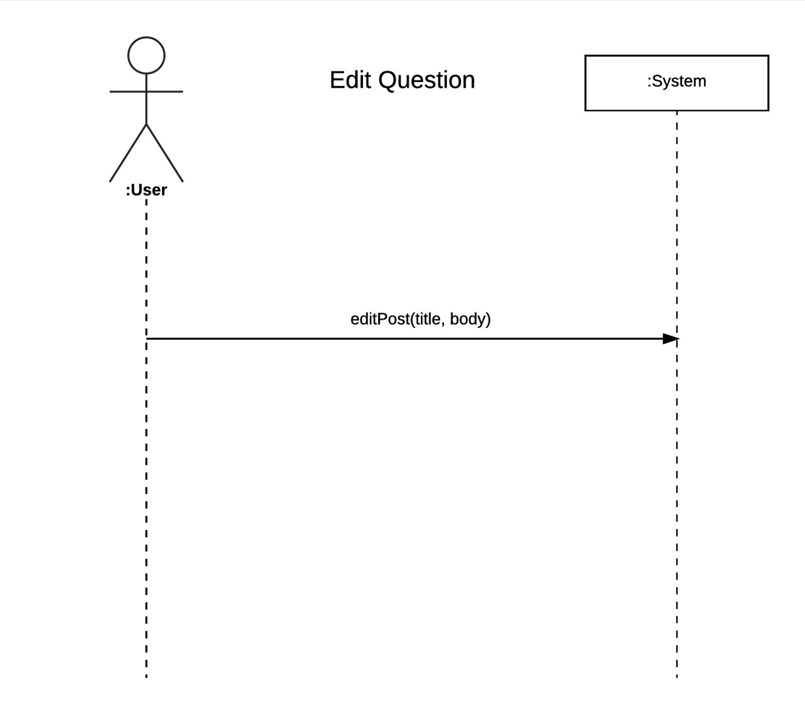
System Sequence Diagrams



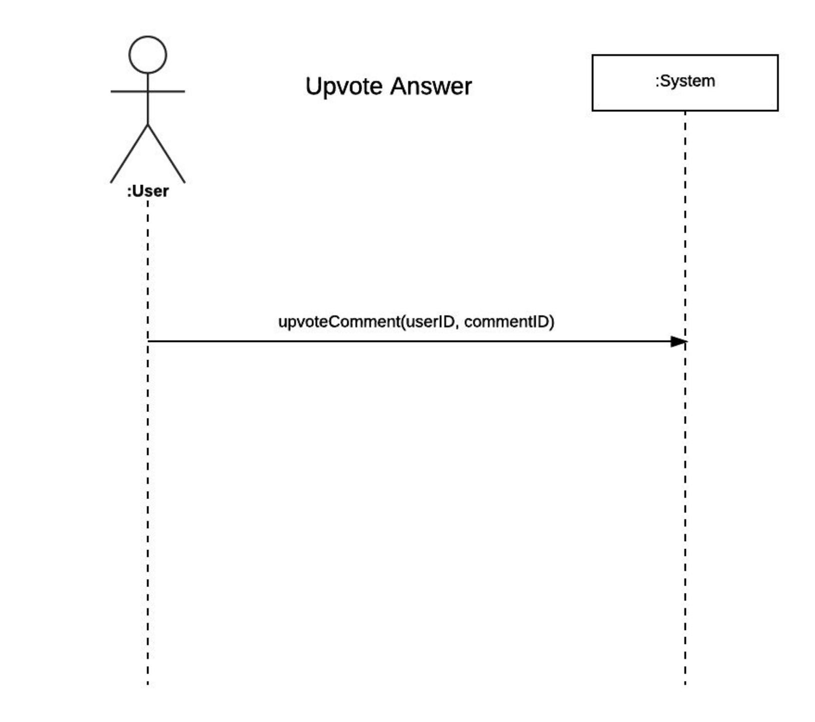
System Sequence Diagrams



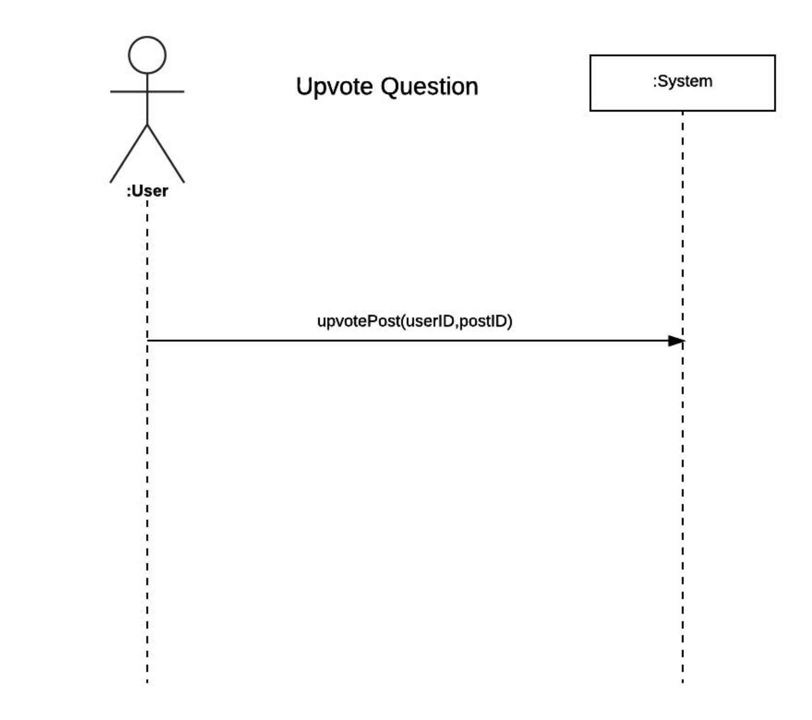
System Sequence Diagrams



System Sequence Diagrams



System Sequence Diagrams



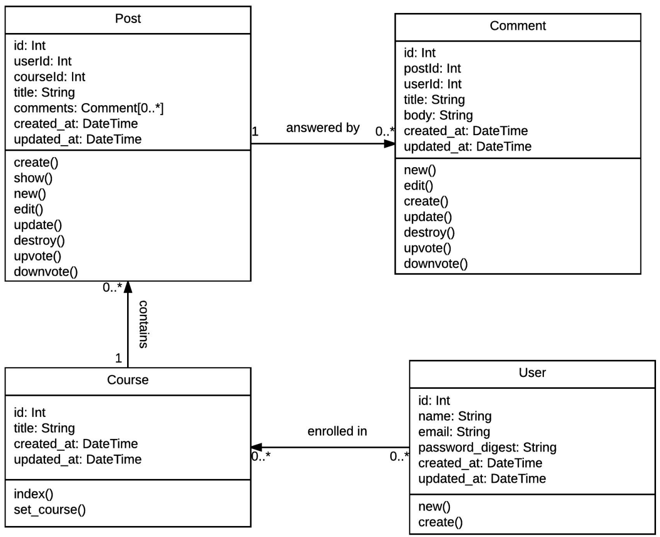
Design

Design: Overview

- Added User class so that specific users have access to specific pages/data. Using the bcrypt gem to encrypt/decrypt user passwords.
- Added a Courses table to keep track of available courses in NKU Net.
- Added a CoursesUsers table bridge table which allows users to register for courses. Each user has many courses and each course has many users.
- User's can upvote/downvote both questions and answers. So we added a PostVotes and CommentVotes table to keep track of how many upvotes/downvotes a question or answer has.

Design Class Diagram

Design Class Diagram With All Classes & Associations



Post

id: Int
userId: Int
courseId: Int
title: String
comments: Comment[0..*]
created_at: DateTime
updated_at: DateTime

create()
show()
new()
edit()
update()
destroy()
upvote()
downvote()

Comment

id: Int
postId: Int
userId: Int
title: String
body: String
created_at: DateTime
updated_at: DateTime

new()
edit()
create()
update()
destroy()
upvote()
downvote()

Course

id: Int
title: String
created_at: DateTime
updated_at: DateTime

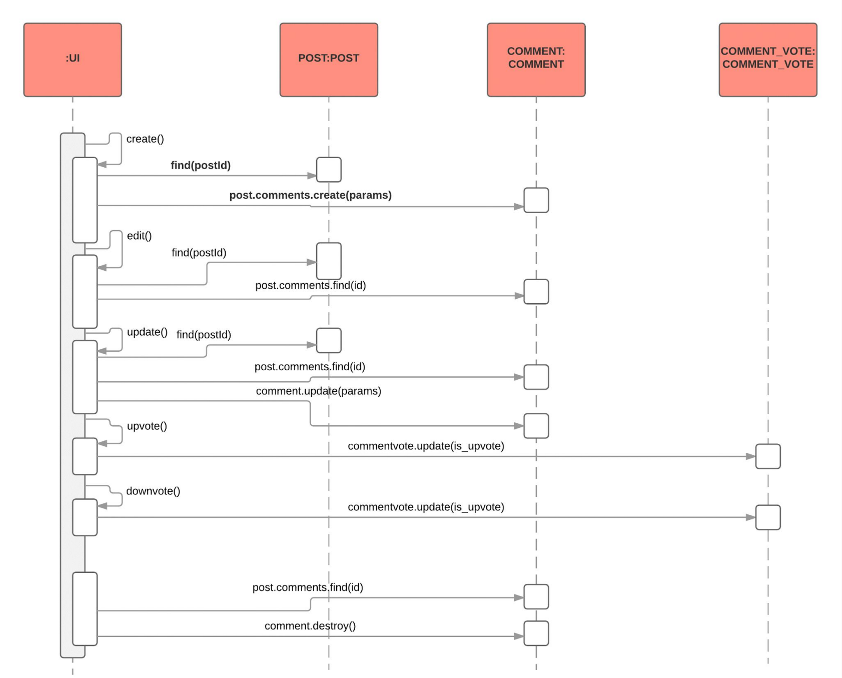
index()
set_course()

User

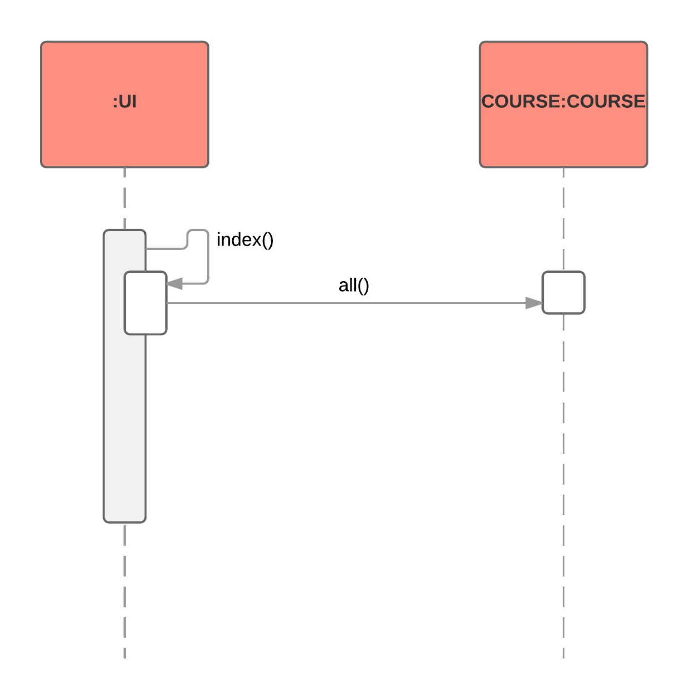
id: Int
name: String
email: String
password_digest: String
created_at: DateTime
updated_at: DateTime

new()
create()

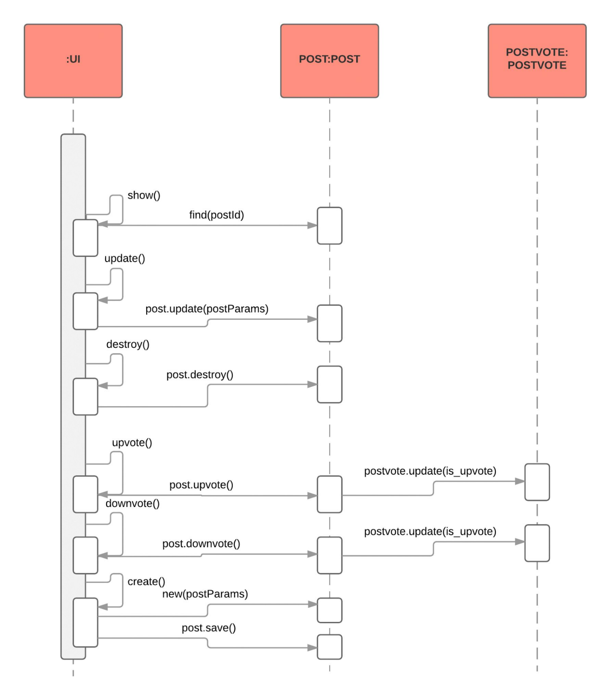
Design Level Sequence Diagrams - Answer



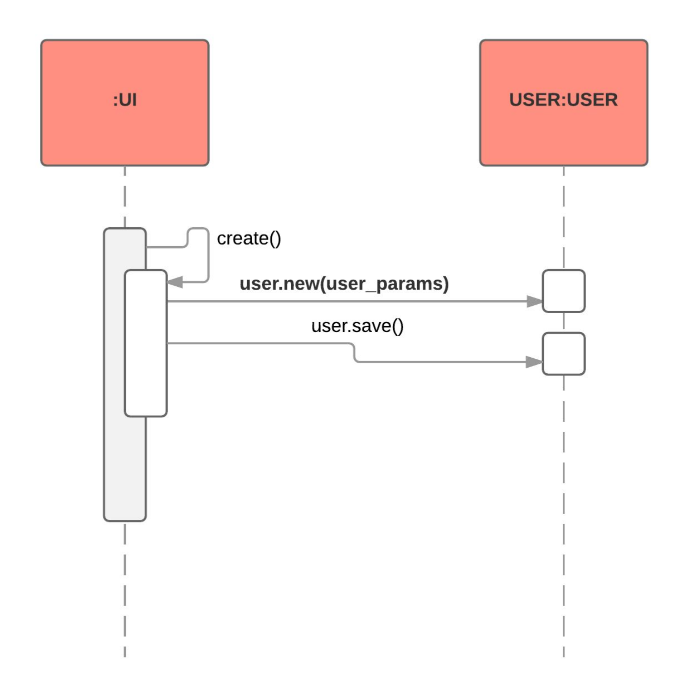
Design Level Sequence Diagrams - Course



Design Level Sequence Diagrams - Question



Design Level Sequence Diagrams - User



Discussion of Design Decisions

- Design decisions for this iteration diverted minimally from original decisions.
- Major UI/Design changes include:
  - more modern look with less gradients
  - Removal of footer
  - added use of Bootstrap UI Components

Discussion of Database Design

Course:

title:	string
created_at:	datetime
updated_at:	datetime

```
$ bin/rails g scaffold Course title:string
```

User:

name:	string
email:	text
password_digest:	integer
created_at:	datetime
updated_at:	datetime

```
$ bin/rails g model User name:string
email:text password_digest:string
```

Discussion of Database Design

PostVotes:

post:	references
user:	references
is_upvote:	boolean
created_at:	datetime
updated_at:	datetime

```
$ bin/rails g model PostVotes post:references
user:references is_upvote:boolean
```

CommentVotes:

comment:	references
user:	references
is_upvote:	boolean
created_at:	datetime
updated_at:	datetime

```
$ bin/rails g model CommentVotes
comment:references user:references
is_upvote:boolean
```

Demo

Any questions?