Design Doc:
Bittiger Communication Project

Author: Fangmin Wei, Jie Zhao
May, 2017

Overview

We aim to create a discussion forum for instructors and students. A discussion Forum is on a video basis rather than on a course basis. When a student has a question for any of a video lecture, he/she can go to just choose that lecture, and go to the discussion forum. He/she can either look for answers from a similar question students have asked before, or he/she can start a new discussion on their own. Once the new discussion is posted successfully, the instructors/mentors will get a notification via email right away, so that they can response to the question as soon as possible. Students can also follow a question.When they do, they will receive notification via emails for any updates of the question. While instructor and mentors will receive notification of updates of any questions through email.

Major Use Cases

1. Students can go to the discussion forum under a video that he/she is watching.
2. Students can search or filter the question list in order to find a similar question to his/her.
3. Students can follow a question. Once he/she follow a question, he/she will get a notification via email every time when the question has any updates.
4. Students can add answers to a question, or reply to others' answers.
5. Students can start a discussion. Once the new discussion is posted successfully, the instructors or mentors will get a notification email, so that student's question can receive a response from the instructors/mentor as soon as possible.

High-level Stack Diagram

Stack	Technologies
Client	React.js
Server	Node.js, Redis, MongoDB

Detailed Design

Client-side

Schema

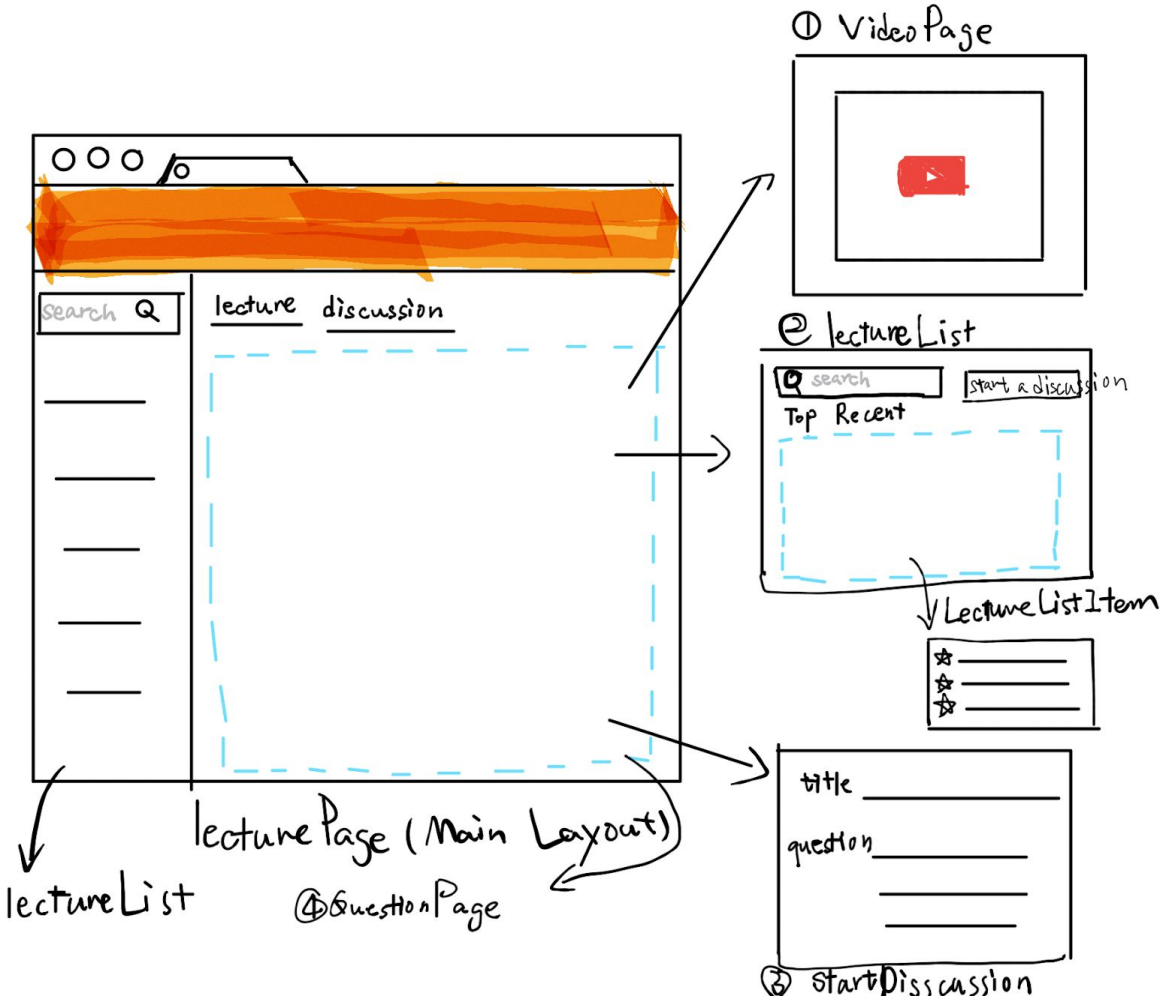
We use React.js to build frontend components - Lecture, Discussion, Question.

Container Component and Representatioin Component

To start, we have to break the rule of mixing 'behavior' with 'how to render the view' --the two things that should stay separate.
so we always create two types of components. the two component types will be conceptually called Container Components and Presentational Components, a.k.a "smart" and "dumb" components.

in short, Container Components source the data and deal with state. State is then passed to Presentational Components as props and is then rendered into views.

we create many pairs of “smart” and “dumb” components:



LecturePage component(the “smart” component) is the ancestor of all the components.It has a child, call LectureList, who take up a fixed space. LecturePage also has other 4 children, they are called: VideoPage, QuestionList, StartDiscussion and QuestionPage. QuestionList has its own child, QuestionListItem. QuestionPage has a child call QuestionPageFirstLevelReply, who has a child called QuestionPageSecondaryReply. Each time, only once child component of LecturePage component is rendered.

before we refactor the project to Redux. We need to just use “smart” and “dumb” components scheme to manage our data more properly, or to put it in another way,in a decoupling way.

React-Router

React Router's power come in when we use multiple routers to define which component should render based on which path is current active.

```
export default (
    <Route path="/" component={Nav}>
        <IndexRoute component={App} />
        <Route path="/home" component={HomeCourses} />
        <Route path="/profile/:userId" component={CourseList} />
        <Route path="/lecture/:courseId/:videoId" component={LecturePage}>
                <IndexRoute component={LectureVideo} />
                <Route path="/:courseId/discussion/:videoId" component={QuestionList} />
                <Route path="/:courseId/:videoId/question/:questionId" component={QuestionPage}/>
                <Route path="/:courseId/:videoId/start_discussion" component={StartDiscussion}/>
            </Route>
        <Route path="/logout" component={LoginPage} />
    </Route>
)
```

As the pic shown above. LecturePage is the ancestor component after we click the ebooks.

Main API

Client side provides APIs to get and set data

API	Description
Auth.authenticateUser(email) Auth.isUserAuthenticated() Auth.deauthenticateUser() Auth.getEmail()	Authentication related methods.
LoginPage.processForm() LoginPage.changeUser()	Get and process user input on the login form. Login user when credentials meet those in database.
CourseList.LoadUserCourses()	Get Courses slected for logined user when user clicks profile button
LecturePage.getLectureVideoList()	Get lecture list and video list for the course user clicks. Loading first lecture and first video by default.
QuestionList.getQuestionList()	Get Question list from server when user clicks discussion tab.
QuestionListItem.followQuestion()	When user click the star, increase the followers number, and change to different color star.
QuestionPage.getQuestionDetail()	When user click a specific question, get the question detail including content, how many answers, etc.
QuestionPage.followQuestion()	Follows the answer to question on the detail page when user clicks star.
QuestionPage.addNewAnswer()	Post new answer to the question when

QuestionPageFirstLevelReply.followQuestion()	Follows the reply to answer when user click the stat.
StartDiscussion.postNewDiscussion()	Post new discussion when user click new Discussion button.

When user hit tab ‘discussion’, QuestionList component is the next to be displayed. Once receive response data of question list from server, we set the question list to the state. Due to that

Extended Functionalities

Add filter for news categories.
Add search bar to search for keywords in question title.

Server-side

Schema

We have 1 route file taking care of client-side requests, in the callback function, we will call API from service which handles CRUD of the mongodb database.

Services

here are four services:
course service which handles with requests getting courses user bought or subscribed to.
lecture service which handles with requests getting lecture list, all that request information.
discussion service which handles with requests getting/posting question list
answer services which handles with requests getting/posting new answers or replies.

Database

Resources

MongoDB

Profile

The schema of a profile:

Name	Type	Description
userId	string	userId to identify an user.

courseId	string	listed all courseId for courses chosen by user

Course

The schema of a course:

Name	Type	Description
courseId	string	courseId to identify a course.
title	string	The title of this course.
desc	string	Detailed description of this course.
free_lec	string	Mark to set this course a free lecture or not.
duration	string	Time length for this course.

CourseLecture

The schema of a courseLecture:

Name	Sub	Type	Description
courseId		string	courseId to identify a course.
name		string	The title of this course.
lectureList	title	string	The title of this lecture.
	videoId	number	videoId to identify the video linked to current lecture.
	url	string	Url link to linked video of this lecture.

Discussion

Name	Sub	Type	Description
courseId		string	courseId to identify a course.
videoId		number	videoId to identify a video.

questionList	title	string	The title of this question.
	author	string	videoId to identify the video linked to current lecture.
	date	string	Url link to linked video of this lecture.
	follow	number	Shows how many follower of the question
	replies	number	Shows how many people reply to the question
	id	number	Id to identify the question

Answer

Name	Sub	Sub-sub	Type	Description
courseId			string	courseId to identify a course.
videoId			number	videoId to identify a video.
questionId			number	questionId to identify a question.
title			string	The title of this question.
date			string	Creation date of this question.
author			string	The author of this question.
content			string	The content of this question.
answers	username		string	The author of this answer.
	email		string	videoId to identify the video linked to current lecture.
	date		string	Url link to linked video of this lecture.
	mentor		boolean	Specifies if the author of this answer is a mentor or not.
	upvote		number	Shows how many people upvotes the answer.
	content		string	The content of this answer.
	reply	username	string	The author of this reply.
		date	string	Creation date of this reply.

		mentor	boolean	Specifies if the author of this reply is a mentor or not.
		upvote	number	Shows how many people upvotes the reply.
		content	string	The content of this reply to the answer.

Todo

When user login, we read the lecture list and question list from database, and keep in Redis.

Scalability (TBD)

Test Plan (TBD)

Future Work

See “todo”s in each section

Reference (TBD)