Enhancement of Bodhitree through the

Development of Additional Features

Avijeet Gaikwad
143050101

Guided by
Prof. Kameswari Chebrolu

Department of Computer Science and Engineering
IIT Bombay

27th June, 2016

Outline

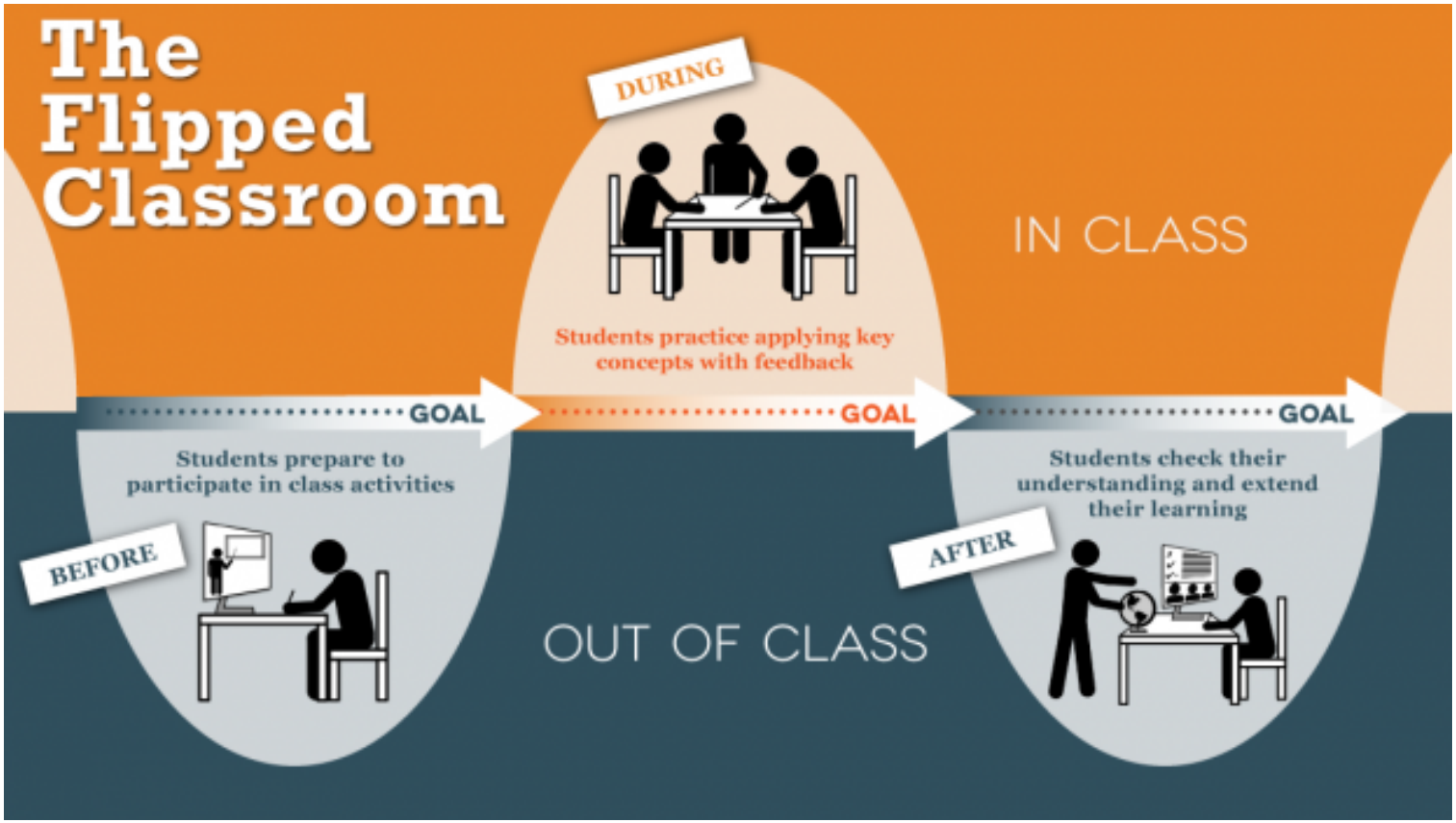
Bodhitree

Introduction and Details

- Bodhitree is an e-learning platform
  - Design to mimic classroom setting
  - Host multimedia textbooks
- Types of users:
  - Instructor
  - Student
- Types of courses
  - Bodhi-class
  - Bodhi-book

Bodhi-class

A flipped classroom model



Source:[1] The University of Texas at Austin: Faculty Innovation Center

Figure : Flipped classroom model

Bodhi-book

A multimedia textbook model

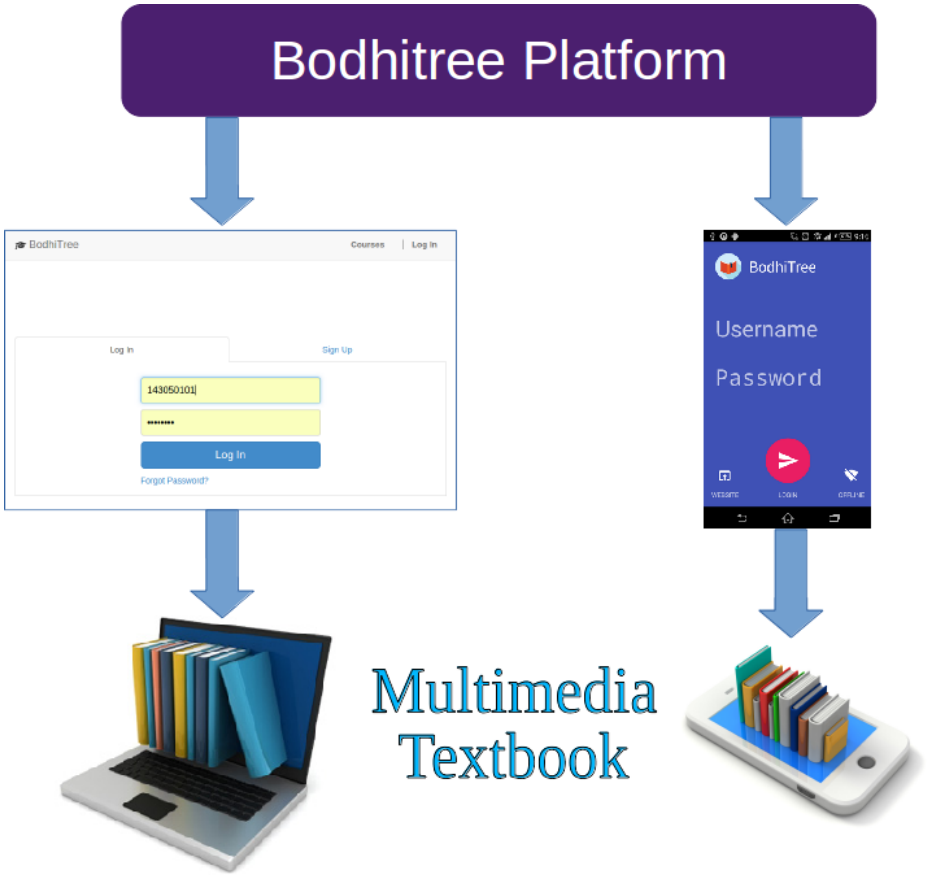


Figure : Multimedia textbook model

Overview of the components that constitute Bodhitree

- Courseware
  - Chapters
  - Concepts
  - Progress
- Learning elements
  - Videos
  - Quiz
    - In-video quiz
    - Out of video quiz
  - Documents
- Interaction
  - Discussion forums
  - Chat
- Labs
  - Report grading
  - Assignments

Development details

- Django web framework (Back-end)
  - (M) Designing the model
  - (V) Creating the templates
  - (C) Writing the views and mapping the URLs
- React.js (Front-end)
  - Component based design
  - Virtual DOM

Objective and Motivation

- Objective is to enhance the Bodhitree platform through the addition of new features
- Need for new features, development ideas:
  - Instructor requirements
  - Student's and Instructor's feedback
  - Brainstorming sessions
- Classification of features pertaining to:
  - Bodhi-class
  - Bodhi-flipped
  - General

Classification and Overview of the work

- Bodhi-class
  - Marks for offline exams
  - Archiving emails
- Bodhi-book
  - Prerequisites and course graph
  - Access control
- General
  - Facebook style notifications
  - Miscellaneous
    - In-video quizzes ON/OFF
    - Setting importance to threads
    - Email instructors when a new thread is added

Marks for offline examinations

Problem Statement

- Several offline exams are conducted in the Bodhi-class model
- The marks for these offline exams are usually sent by emails, having CSV files attachments
- It is required to have an interface to display these marks to the students on Bodhitree itself

Specifications

Instructor Specifications

- Upload a CSV containing the students marks
- The file should have the exam details and marks, along with the usernames of the students, in the following format

Students Specifications

- Students must be able to view only their marks that are uploaded for the course

User Interface as seen by the instructor

Upload Marks

Marks File (csv) Choose File No file chosen

Upload

Student	Q1 (20)	Midsem (30)	Q2 (20)	Endsem (30)
root	18	25	12	22
avi	12	20	14	24

Figure : Instructor's view of CSV upload and display of students marks

User Interface as seen by the student

Q1 (20)	Midsem (30)	Q2 (20)	Endsem (30)
12	20	14	24

Figure : Student 2's view of his marks

Uploading CSV file

EXAM_TYPE	Q1	Midsem	Q2	Endsem
MAX_MARKS	20	30	20	30
Student 1	18	25	12	22
Student 2	12	20	14	24

Storing marks in the database

id	course_id	exam_type	max_marks
1	2	Q1	20
2	2	Midsem	30
3	2	Q2	20
4	2	Endsem	30

Exams details

id	marksheader_id	student_id	marks
1	1	1	18
2	2	1	25
3	3	1	12
4	4	1	22
5	1	10	12

Marks obtained by students

Figure : Data view of the marks module

Archiving of emails

Problem Statement

- Instructors frequently send emails in the Bodhi-class
- These emails are concerned with important announcements and updates in the course
- No easy way for the instructor to view the emails
- Need for archiving and course-wise categorization

Specifications

- All the emails sent by the instructors must be archived
- Instructor should be able to view all the emails sent by him in a particular course
- Student must be able to view all the emails that are received by him
- Sorting and searching features should be available

Archiving the emails

- Function *archive_mail()* called whenever an email is sent
- Foreign keys to the current *user*, *course* and *list of recipients* (many to many relation)
- Postgres automatically adds the current date and time

Displaying the archived emails

- User makes an AJAX request to the server
- Server checks the dtails of the user in the current session
- Data is fetched in JSON format

```
GET /email_archive/api/getEmails/1/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "emails": [
        {
            "body": "<p>test</p>",
            "recipients_list": "['avijeet7@gmail.com', 'avijeet@cse.iitb.ac.in']",
            "sender_id": 1,
            "cc_list": "[]",
            "from_email": "instructor_CS224:_Computer_Networks@bodhitree.cse.iitb.ac.in",
            "date_sent": "2016-06-14T15:19:06.564Z",
            "reply_to": "avijeetjob@gmail.com",
            "id": 1,
            "course_id": 1,
            "subject": "test"
        }
    ],
    "mode": "S"
}
```

UI for viewing the emails

- Component based design using React.js
- Sorted by most recent first

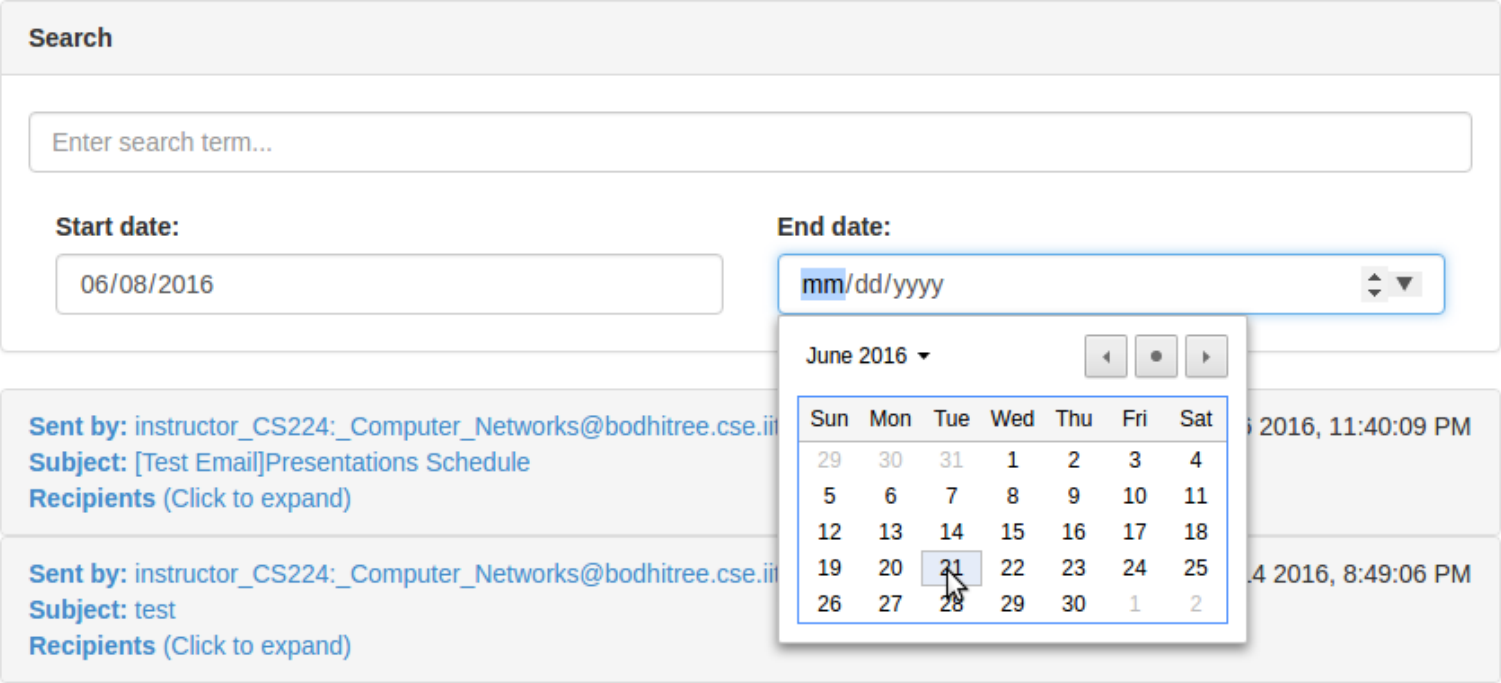
Sent by: instructor_CS224:_Computer_Networks@bodhitree.cse.iitb.ac.in Thu Jun 16 2016, 11:40:09 PM
Subject: [Test Email]Presentations Schedule
Recipients (Click to expand)
avijeet7@gmail.com avijeetjob@gmail.com avijeet@cse.iitb.ac.in

This is a test **email.**

Sent by: instructor_CS224:_Computer_Networks@bodhitree.cse.iitb.ac.in Tue Jun 14 2016, 8:49:06 PM
Subject: test
Recipients (Click to expand)

Search and filter functions

- Searchbox and date filter
- *Instant search*



Prerequisites and course graph

Problem Statement

- Bodhi-book model lacks significant involvement of instructors
- Dependencies in chapters and concepts
- Proficiency levels required for complete understanding
- Guidelines are necessary for students
- Graphical representation of the course and the guidelines help the students to progress in a better way

Specifications

Motivation is to ensure that every student has the required background knowledge before attempting to learn a new concept

- Instructor specifies prerequisites for a concept
- Prerequisites viewable on the concept page
- Graph generated using the course data, which serves as a guideline
- Instructor modifies and finalizes the course graph

Storing the prerequisites

- Concepts/Chapters are the prerequisites for other Concepts/Chapters
- Prerequisites are stored as a list of Concept/Chapter id's
- Default values are a blank list *[ ]*
- In case of a Chapter, a *"g"* character is appended in front of the id
- Example of prerequisites data for a concept:
  ["g1", "5", "6"]

Design

UI to add prerequisites

- Only the instructor has authority
- Concepts categorized into chapters
- Component based design, immediate request

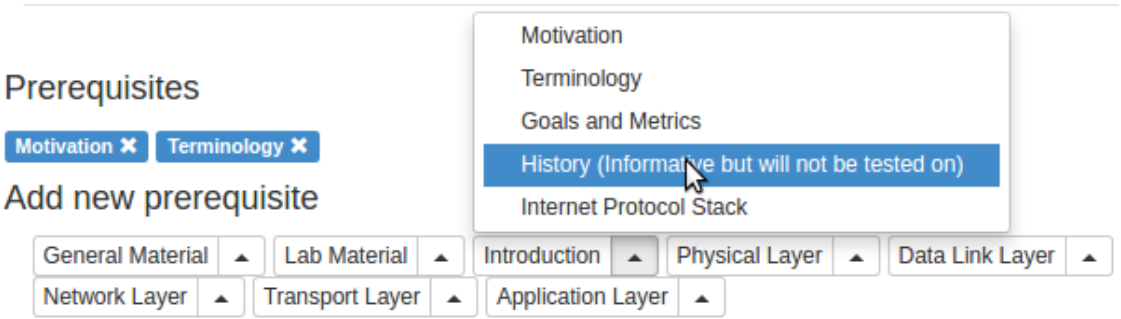
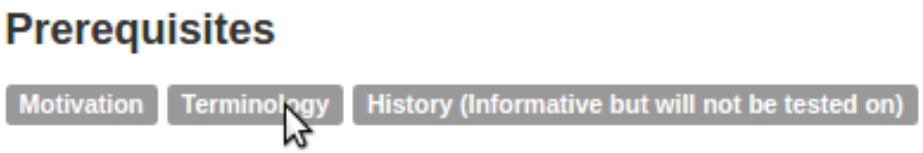


Figure : Adding prerequisite to a concept

Student's view of prerequisites on the concept page



- Navigable links

Design

Creation of the course graph

Several graph generation libraries used to generate initial course graphs:

- **Graphviz:** Python library which generates an image of the final graph, non-interactive
- **Arbor.js:** Javascript library which dynamically generates interactive graphs, allows users to interact with the components
- **Vis.js:** Javascript library, interactive graphs, GUI to create/modify existing graphs

Graph generated using Graphviz library

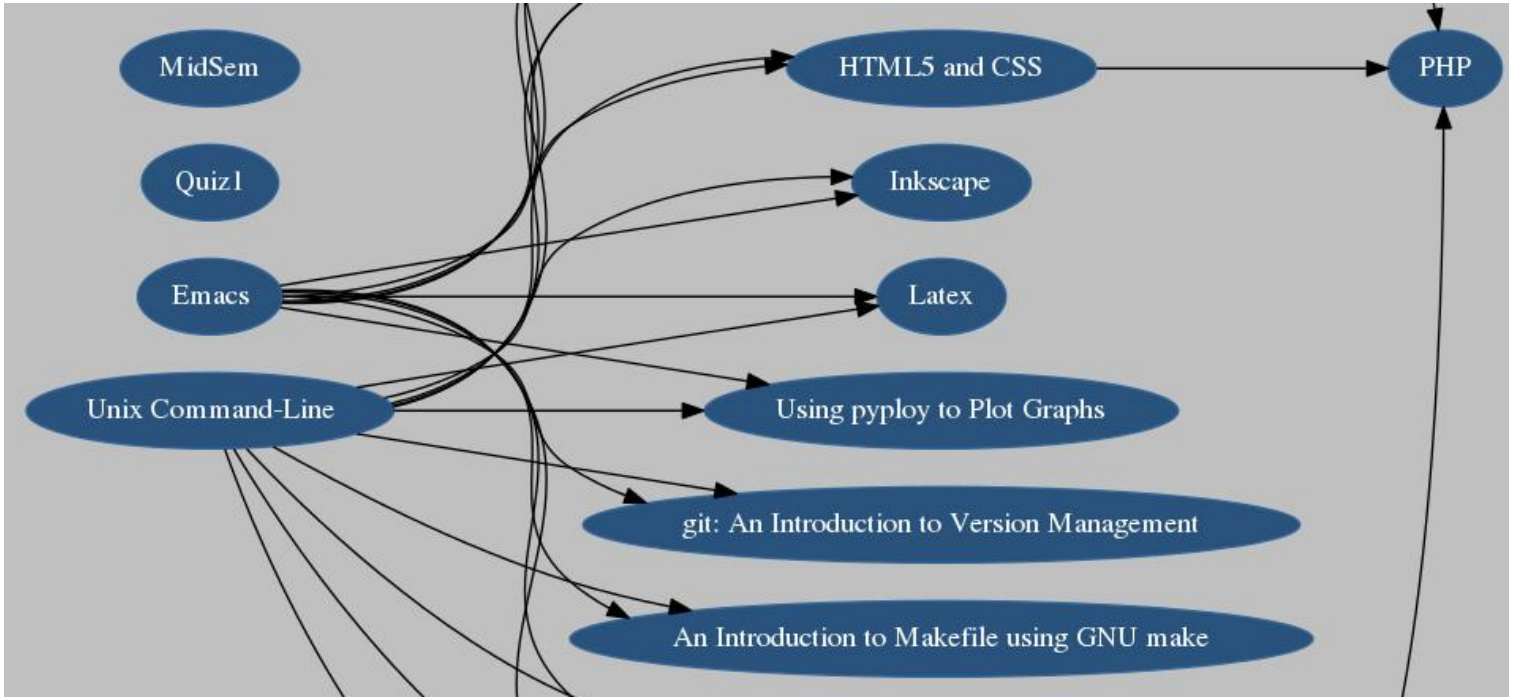


Figure : Part of a complex graph generated using graphviz

Graph generated using Arbor.js

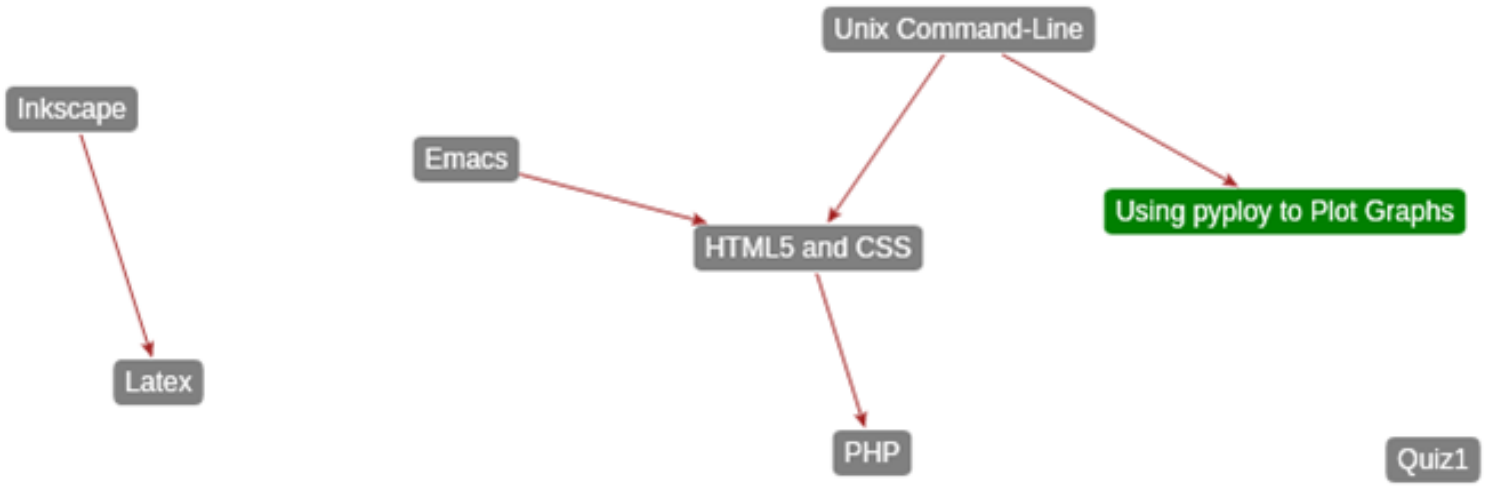
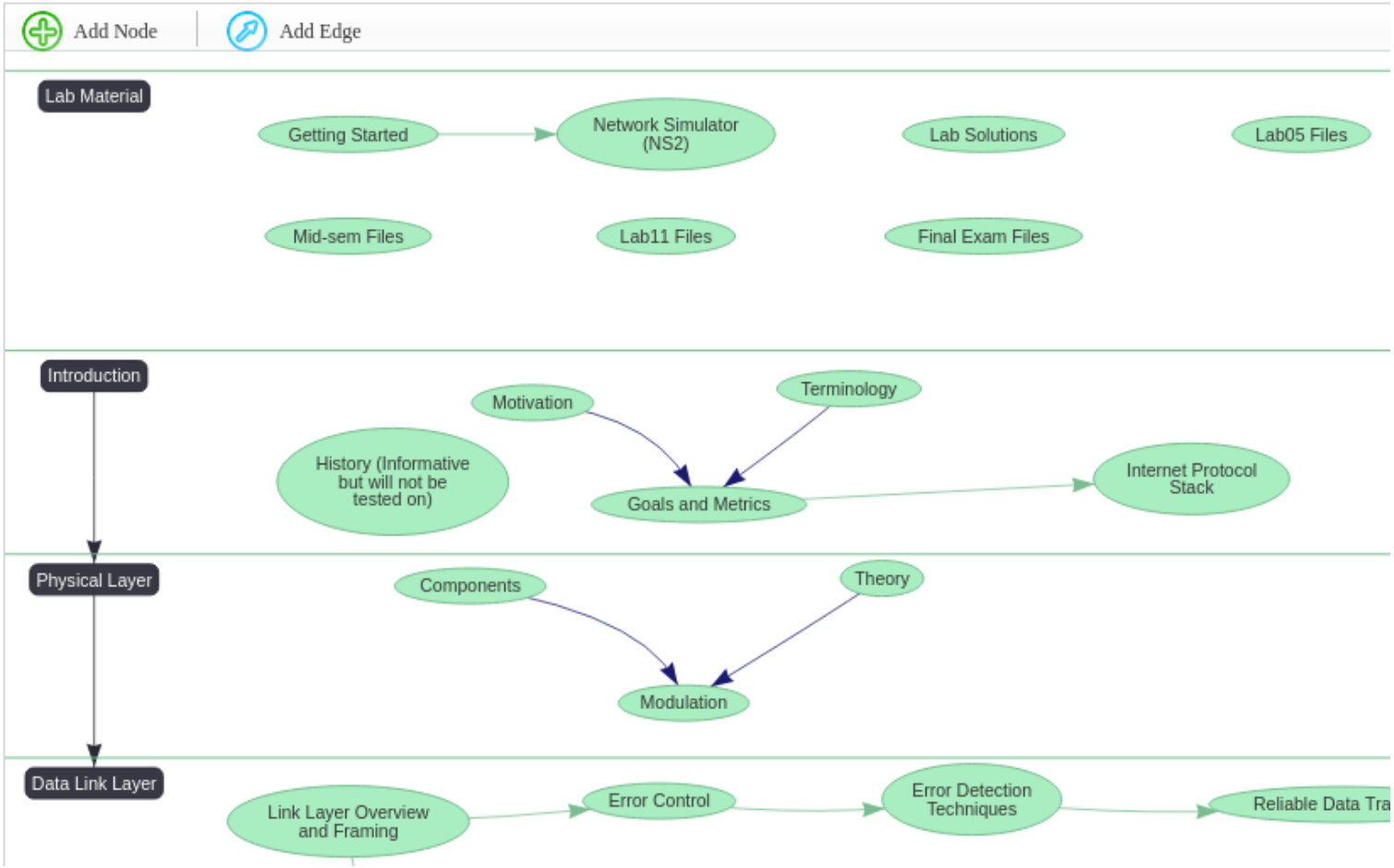


Figure : Part of a graph generated using arbor.js

Graph generated using Vis.js



Design

Creation of graph using vis.js

- Function created to fetch the course data, convert it to JSON and render the graph (html file)
- JSON captured on the client side. Function for creation and organization of nodes and edges written in JavaScript
- Instructor must save the graph before the student can view it
- All the components on the canvas are interactive and modifiable
- Example of Nodes and Edges data:

Nodes	**Edges**
[{id: 1, label: 'Node 1'}, {id: 2, label: 'Node 2'}, {id: 3, label: 'Node 3'}, {id: 4, label: 'Node 4'}, {id: 5, label: 'Node 5'}]	[{from: 1, to: 3}, {from: 1, to: 2}, {from: 2, to: 4}, {from: 2, to: 5}]

Future work

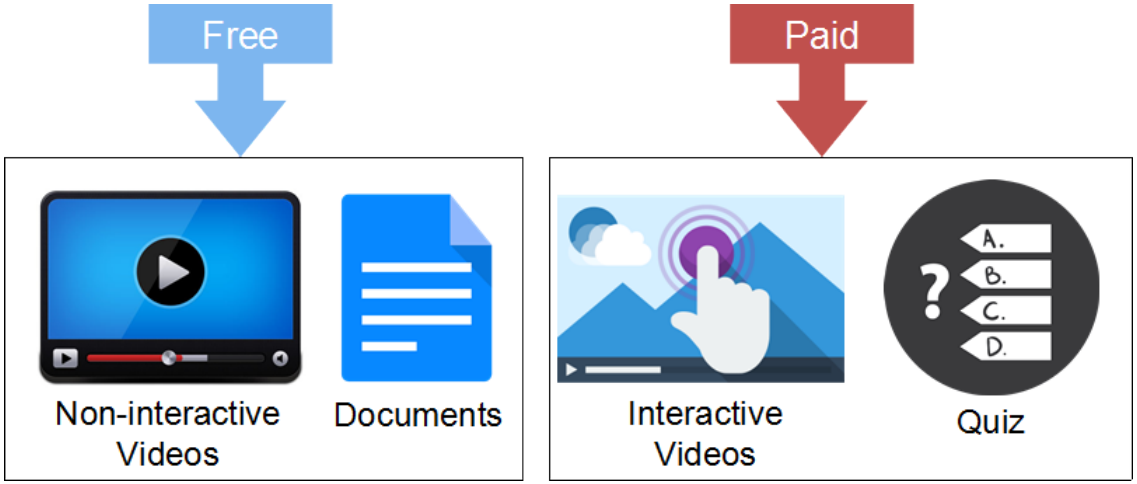
Ideas to empower course graphs

- Using data from the progress module to display proficiency levels of students
- Generating dynamic, student specific graphs for providing alternative learning paths
- Creating multidimensional graphs, which display in-depth information and provide navigation to courseware elements

Access control

Motivation

Providing differential access to users towards the content



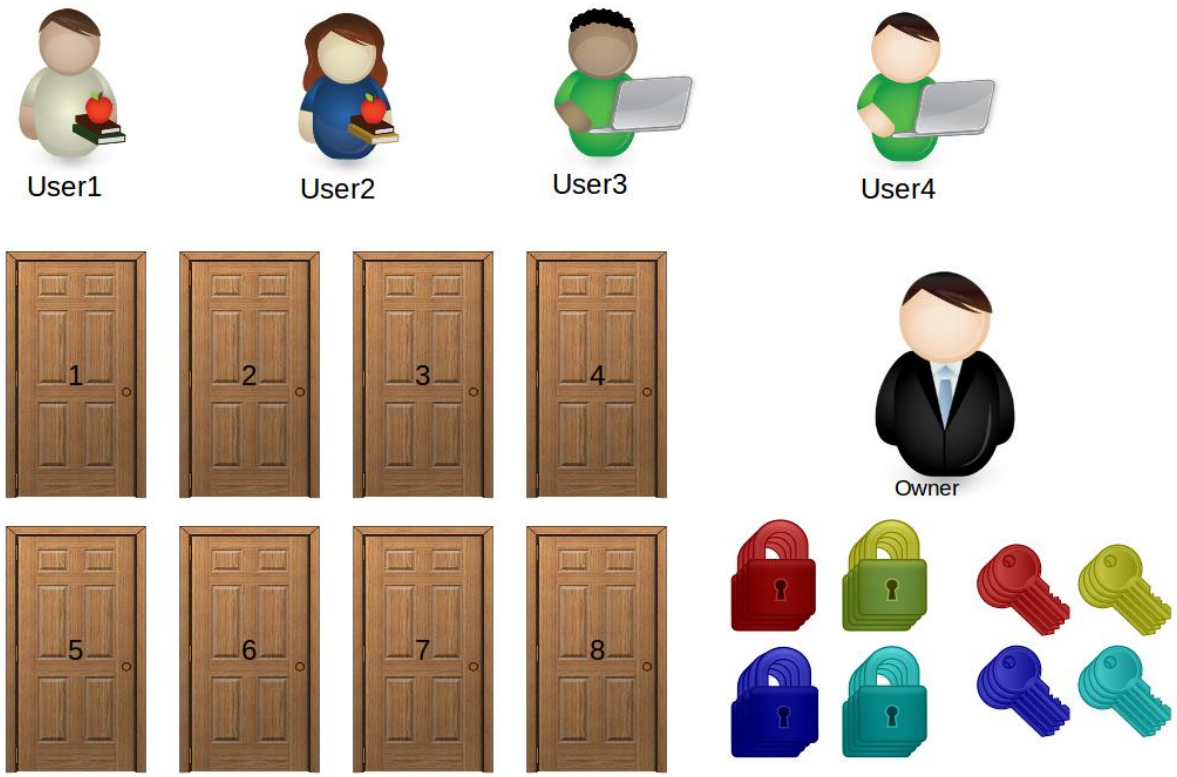
Problem Statement

- Giving the instructor authority to limit the access a student has to the content in his course

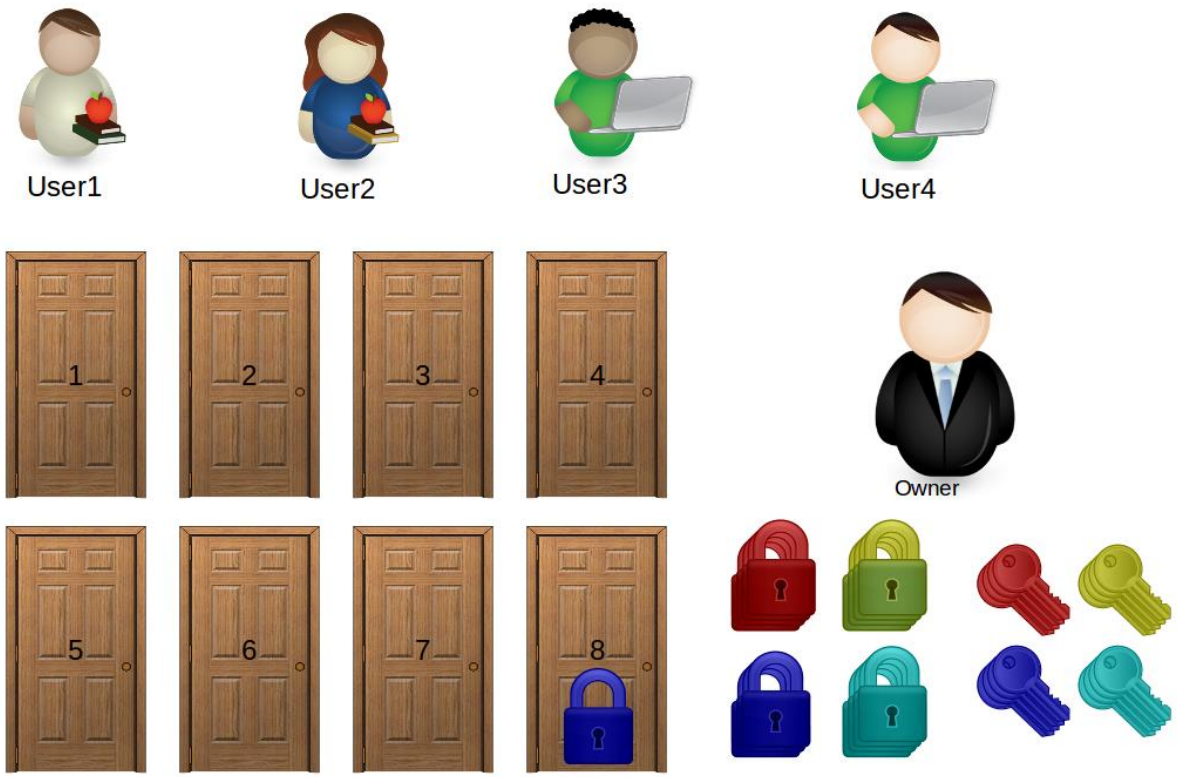
Specifications

Concept of locks and keys

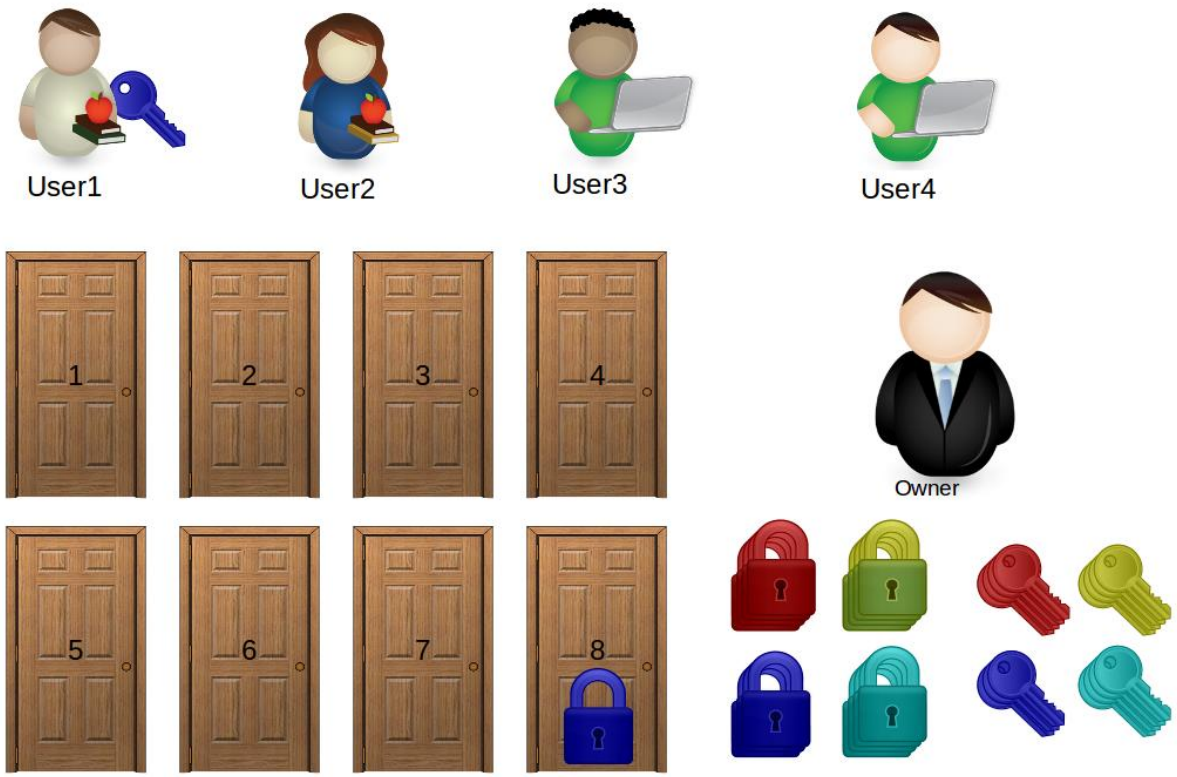
8 doors, 4 users, 1 owner, 4 types of locks and their keys



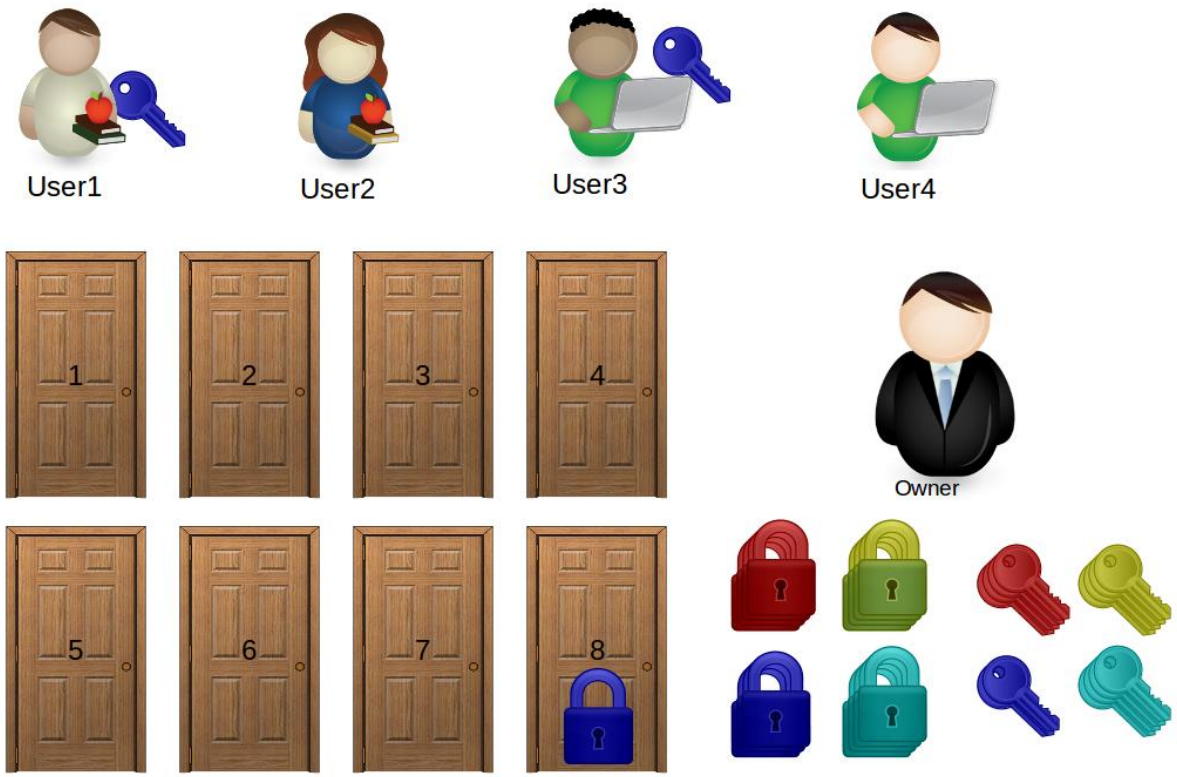
Owner locks door no. 8, thus restricting access to it



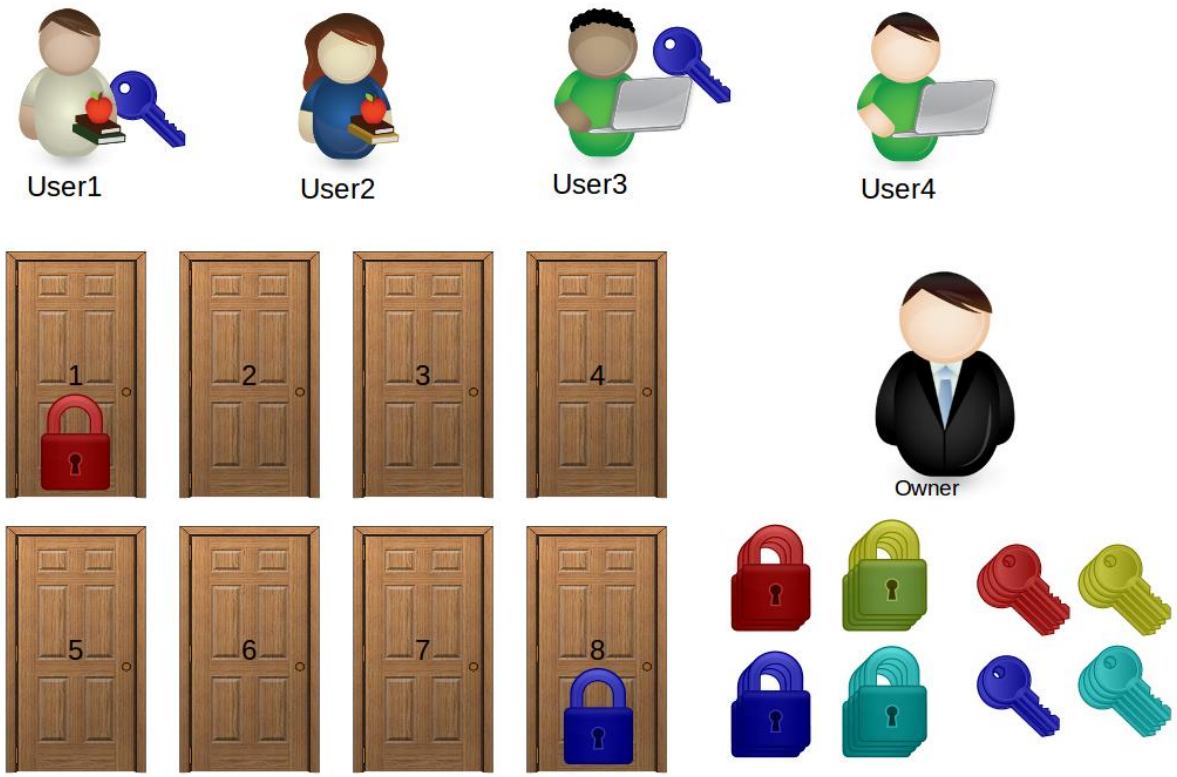
User1 gets access to door 8



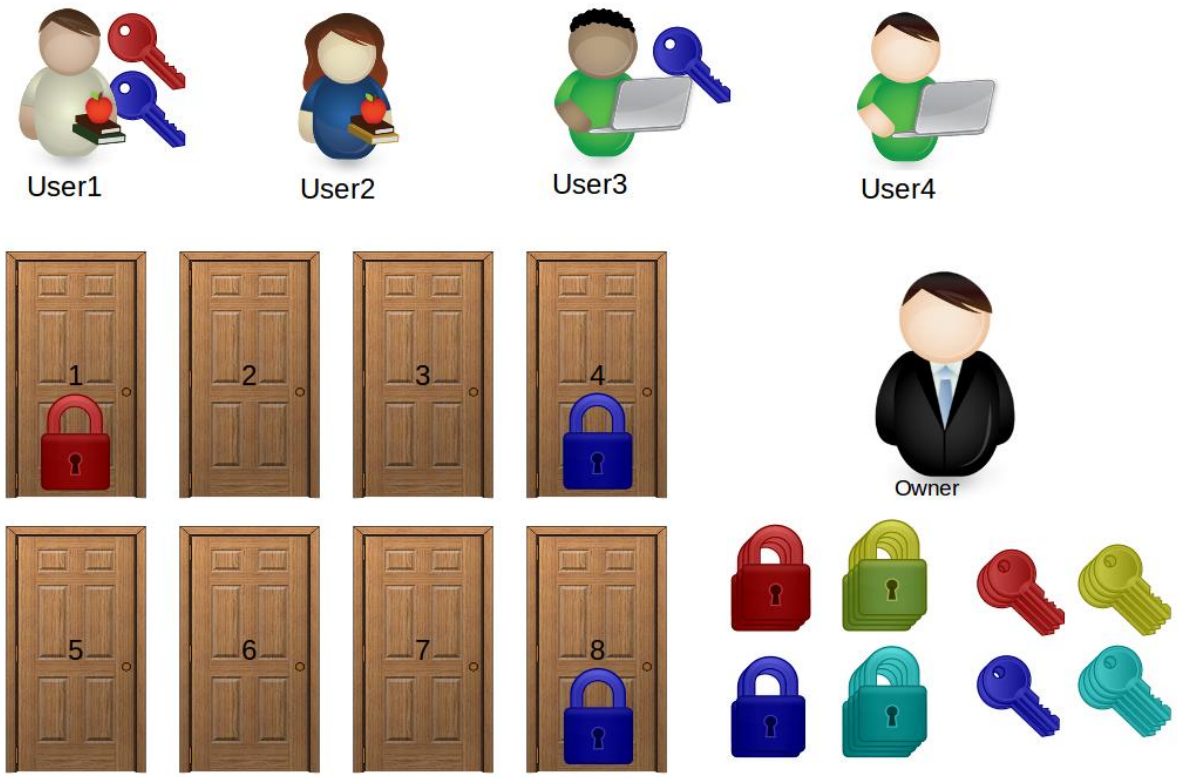
Giving access to other users



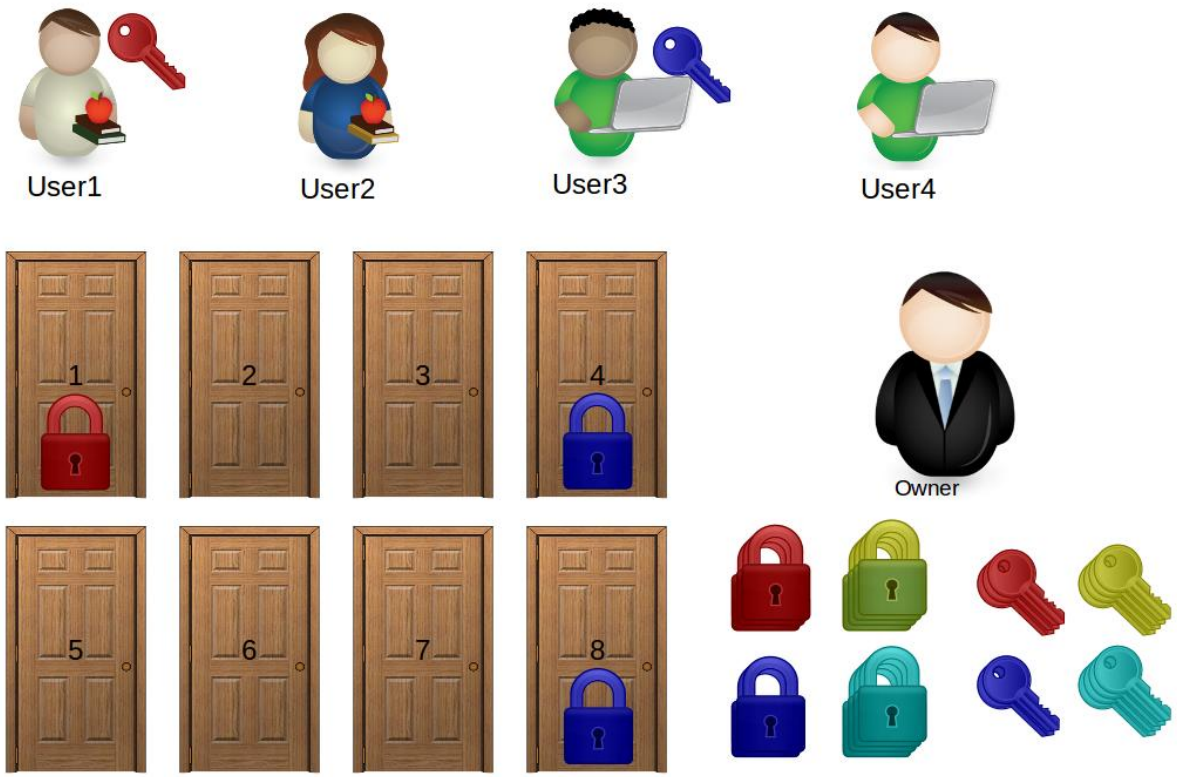
A different type of lock



User1 gets access to multiple doors



Restricting access by taking away the key

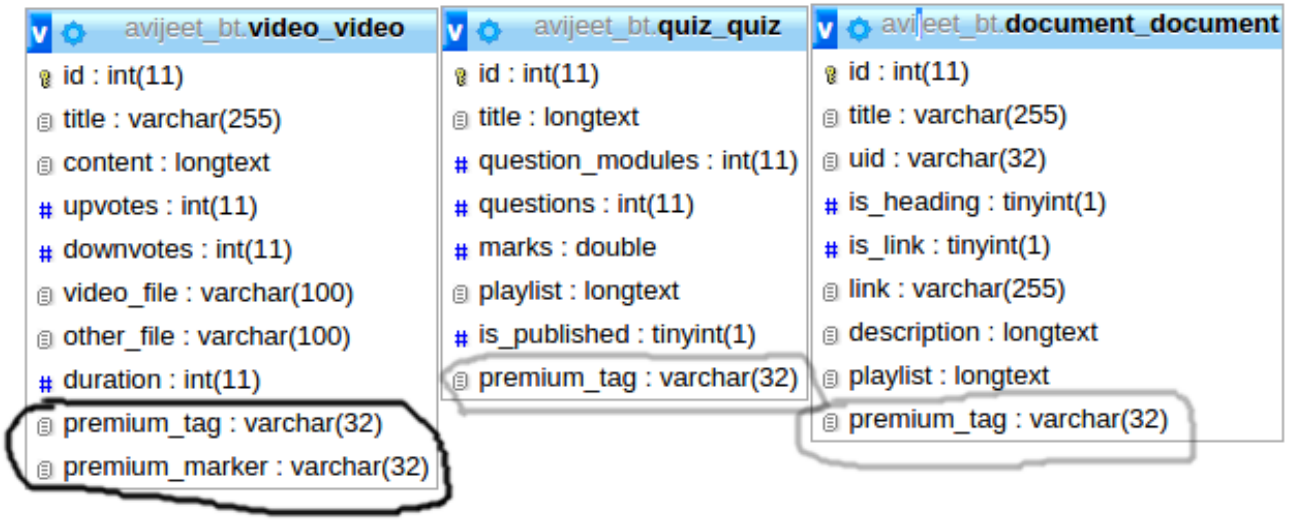


Specifications

- Analogy
  - Owner: Instructor
  - Users: Students.
  - Doors/rooms: Content offered on Bodhitree.
  - Locks: Tagging of content by an instructor.
  - Keys: Granting of access to the students for those tags.
- Learning elements that can be tagged:
  - Videos
  - Documents
  - Quizzes
  - Video markers (In-video Quizzes)

Design
Defaults

- When an element (video, document, quiz or video markers) is created in a concept, the default tag is set as *"Free"*
- A *"Free"* tag denotes that all the students who are registered for that course can access that element
- Fields *premium_tag* and *premium_marker* added in the database



Design

Tagging interface

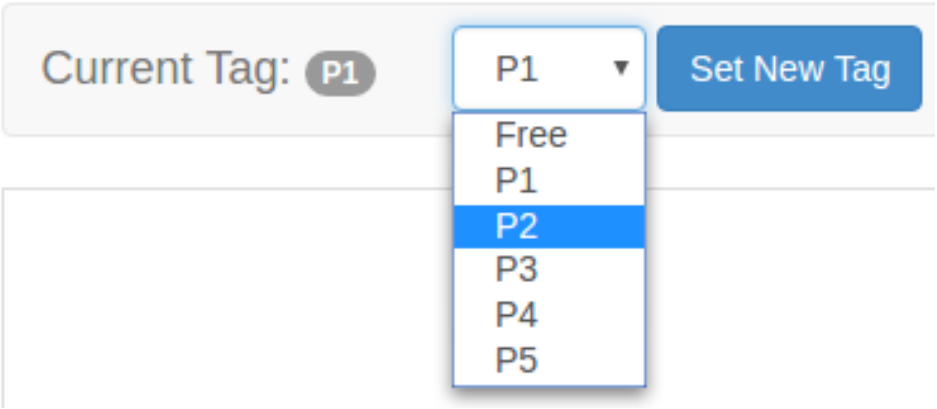
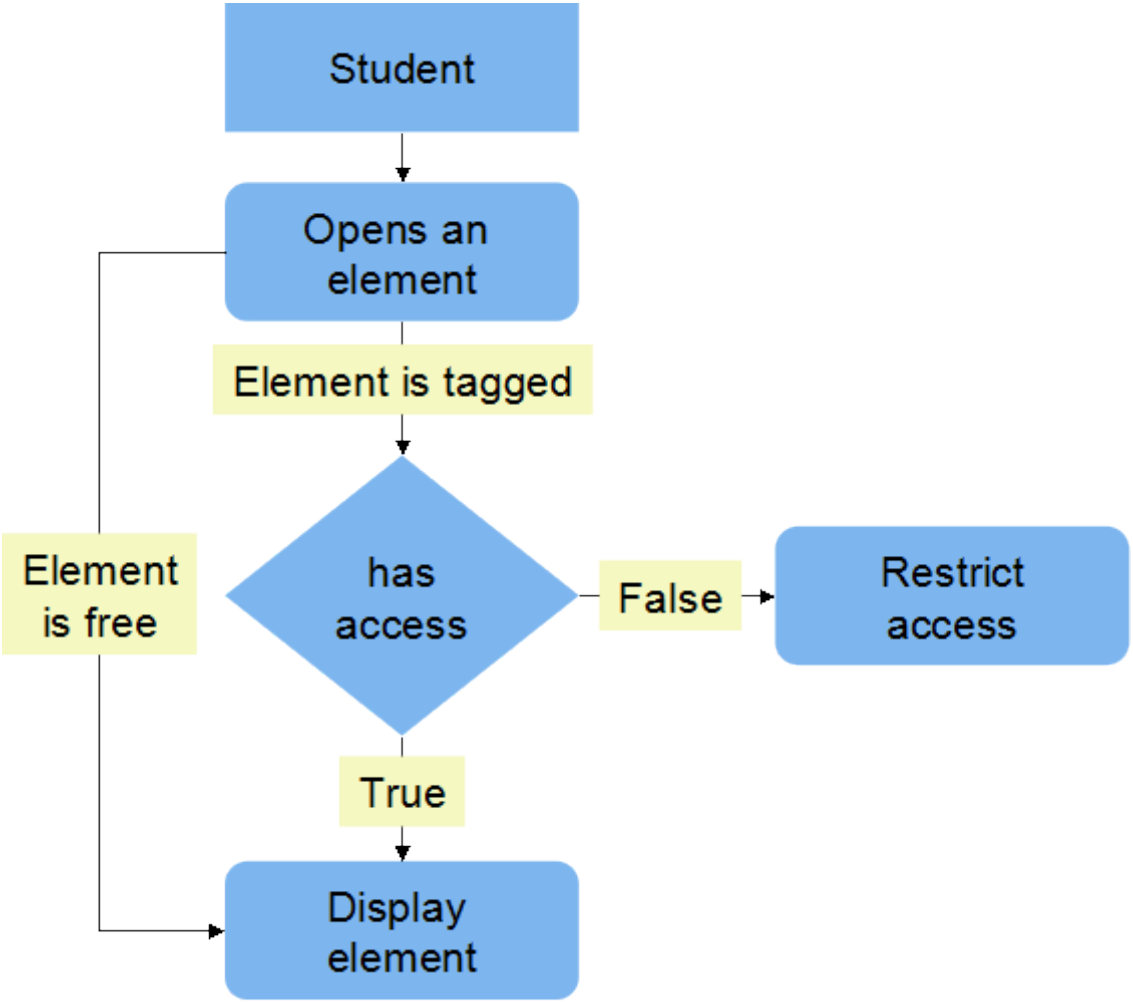


Figure : Tagging interface

- A badge shows the currently set tag
- A drop down box which lists the 5 fixed tags (P1, P2, P3, P4, P5), and the custom tags
- Setting a new tag changes the *premium_tag* field in the database

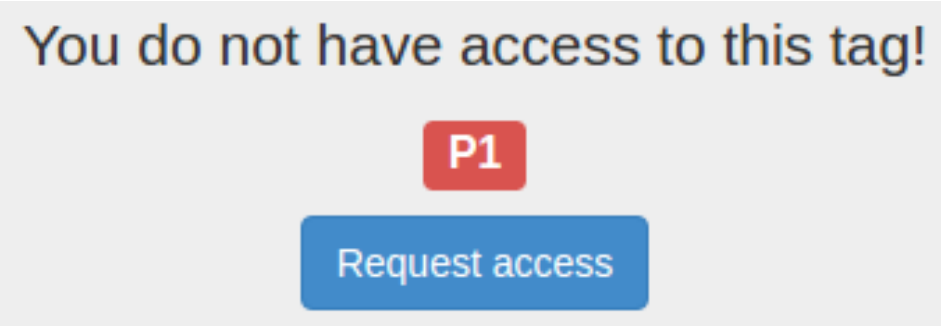
Student's access to a learning element



Design

Student view

- Server check the user request and sends appropriate data
- Data is not sent to unauthorized users
- The field *"has_access"* is set to *false* in the JSON that is fetched, resulting in the following message



Data sent to client who is not having access to an element


```
"content": {
    "id": 6,
    "title": "Components",
    "content": "The video will cover the details of the components that make up the PHY layer.",
    "upvotes": 0,
    "downvotes": 0,
    "video_file": "",
    "markers": [],
    "other_file": "",
    "duration": 0,
    "premium_tag": "P1",
    "premium_marker": "M1"
},
"marker_access": false,
"has_access": false,
"type": "video",
"history": {
```

Design

Granting access to users

- Uploading the CSV file:

stud1	P1	P2	P3
stud2	P1		
stud3	P3		

- Checking username against the registered users
- Storing the tags in the database:

id	user_id	course_id	premium_type
10	1	1	P1
5	1	1	P2
8	1	1	P3
4	1	3	P1

Design

CSV file upload

Interface to upload the CSV file:

Premium users file in csv format:

Choose File No file chosen

Submit

List of users who have access to premium features:

Username	Access Types
root	P1 P2 P3
avi	P1 P2
salman	P2

Design

Upload success

Information displayed after successfully uploading a CSV file

Successfully uploaded the users

Existing premium users:

Username	Access Type
root	P2

Users added as premium:

Username	Access Type
✔ root	P1 P3
✔ avi	P1 P2
✔ salman	P2

Unknown users:

Username	Access Type
? invalid_user	P1

Removing of tags access from users

- Content developer has the authority to remove the tags access
- Checklist provided listing the users and associated tags
- Corresponding entries removed from the database

Interface to remove tag access

List of users who have access to premium features:

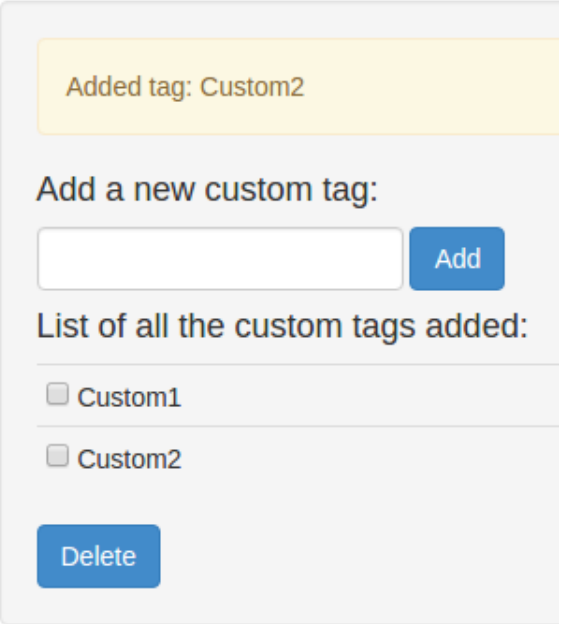
Username	Access Type
root	P2
avi	P1
salman	P2

Delete

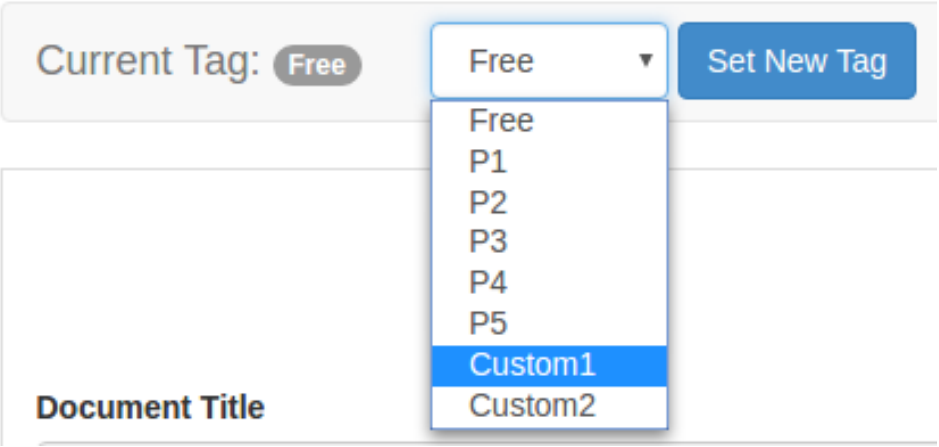
Creation of custom tags by the instructor

- If the 5 default tags **(P1, P2, P3, P4, P5)** are not enough, the instructor may create new tags of his own
- Once created, custom tags ar available in the drop down list of the tagging interface

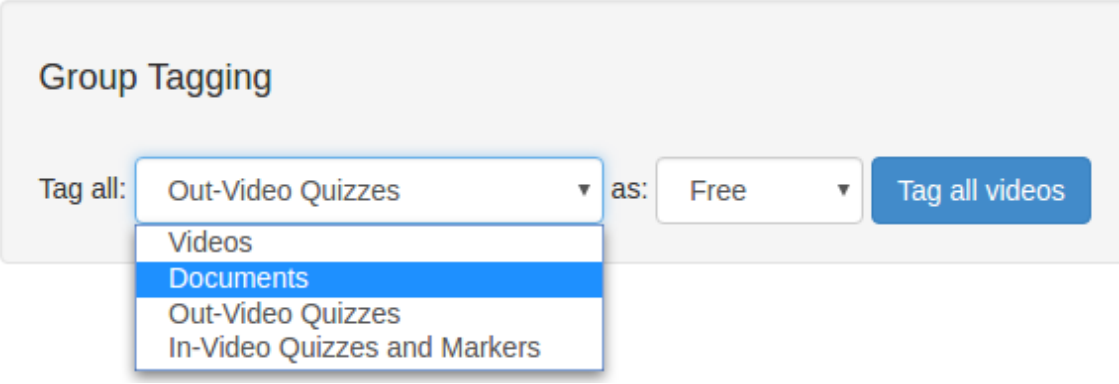
Interface for adding custom tags

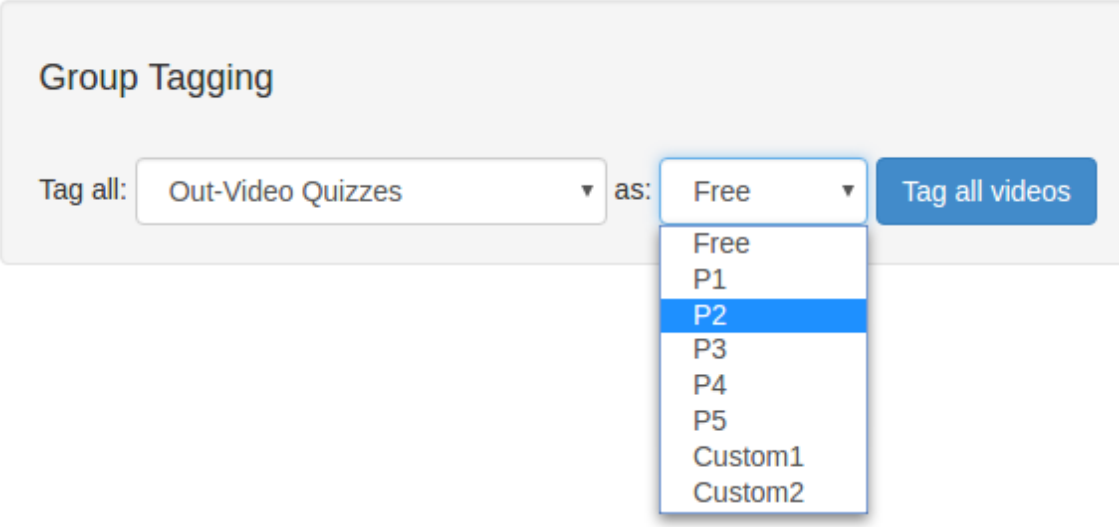


Custom tags available in the tagging interface after creation



Tag all the elements (Videos, Documents, etc.) in one click





Facebook style notifications

Problem Statement

Miscellaneous

- In-Video Quiz ON/OFF
- Setting importance to threads
- Email instructors on addition of a new thread

Conclusion

References I

The University of Texas at Austin: Faculty Innovation Center
https://facultyinnovate.utexas.edu/teaching/flipping-a-class