Bodhitree

E-learning platform development

Avijeet Gaikwad
143050101

Guided by
Prof. Kameswari Chebrolu
and
Prof. Bhaskaran Raman

Department of Computer Science and Engineering
IIT Bombay

October 21, 2015

Outline

Bodhitree Introduction

Components of the platform and introduction

- Small Private Online Courses (SPOCs)
- Bodhitree is an e-learning platform used to host SPOCs
- Components of Bodhitree:
    - Courseware
        - Concepts
        - Discussion Forums
        - Assignments
    - Documents
    - Videos
    - Quizzes

Terminology

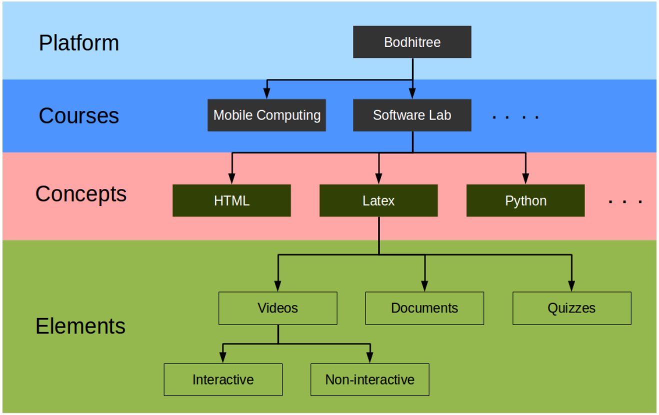
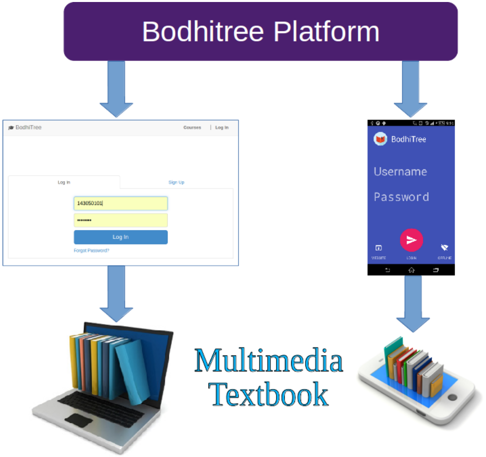


Figure: Components of Bodhitree and terminology

Multimedia Textbook

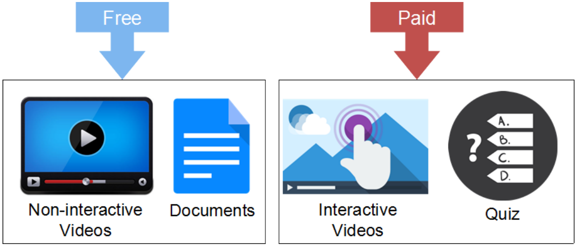


Additional Features

- **Access Control**
- **Prerequisites Graph**
- Discussion section
- Student Progress Tracking
- Miscellaneous:
  - Accounts misuse
  - Server and client side logging
  - Scalability

Access Control

Differential access to content based on payment



Access Control

Problem Statement

- Adding feature to provide paid content
- Privileges set by content developer

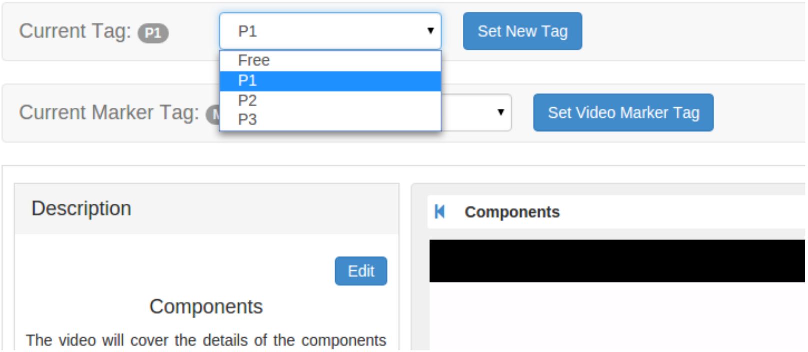
Specifications

- Tagging of concept elements by the content developer, elements being any of the following:
  - Videos
  - Documents
  - Quizzes
  - Video markers
- There are 5 fixed tags *(P1, P2, P3, P4 and P5)*, and the default tag is *Free*
- Content developer can add custom tags
- These tagged elements cannot be accessed by the students having the default *Free* access

Specifications

- Several elements can be tagged under one tag, that will lead to grouping of access
- Tagging video markers
- Content developer gives access to tags to the students
- Students having access to a tag can access all the elements marked by that tag
- Content developer can remove the access to tags

Tagging interface for the content developer



Components

The video will cover the details of the components

Tagging

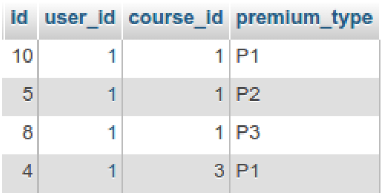
- Tagging interface:
  - A label *"Current Tag:"* followed by a badge
  - A drop down box listing the tags
  - A button labeled *"Set New Tag"*
- Addition of "*premium_tag*" and "*premium_marker*" field

Granting access to users

- Uploading the CSV file:

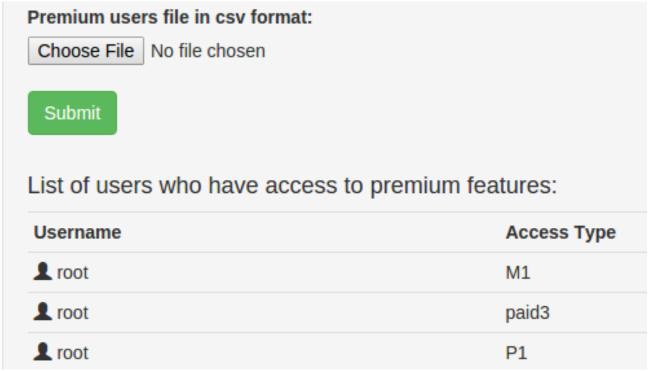
stud1	P1	P2	P3
stud2	P1		
stud3	P3		

- 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

Interface to upload the CSV file:



Premium users file in csv format:

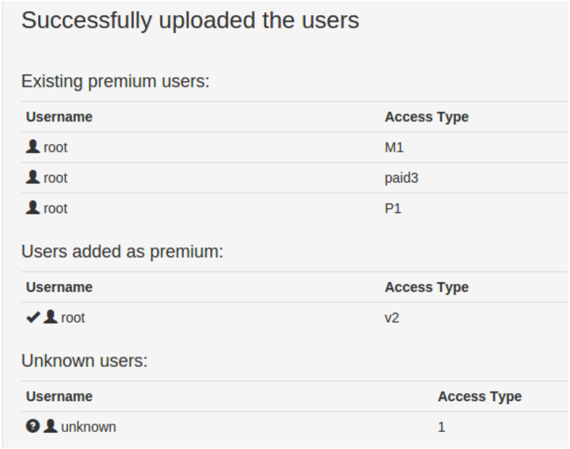
Choose File No file chosen

Submit

List of users who have access to premium features:

Username	Access Type
root	M1
root	paid3
root	P1

Information displayed after an upload



Successfully uploaded the users

Existing premium users:

Username	Access Type
root	M1
root	paid3
root	P1

Users added as premium:

Username	Access Type
✓ root	v2

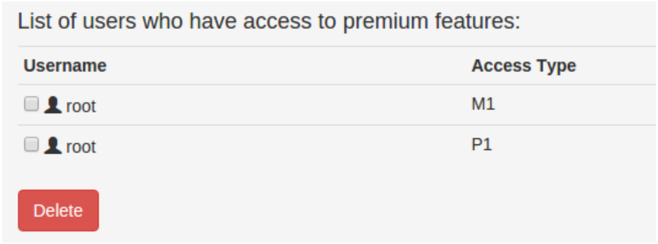
Unknown users:

Username	Access Type
? unknown	1

Removal of access to tags 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

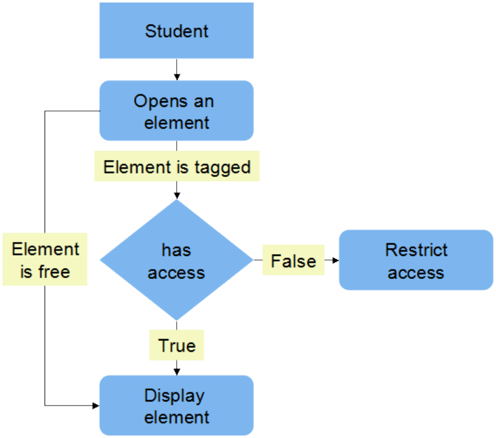
Interface to remove tag access



List of users who have access to premium features:

Username	Access Type
☐ root	M1
☐ root	P1

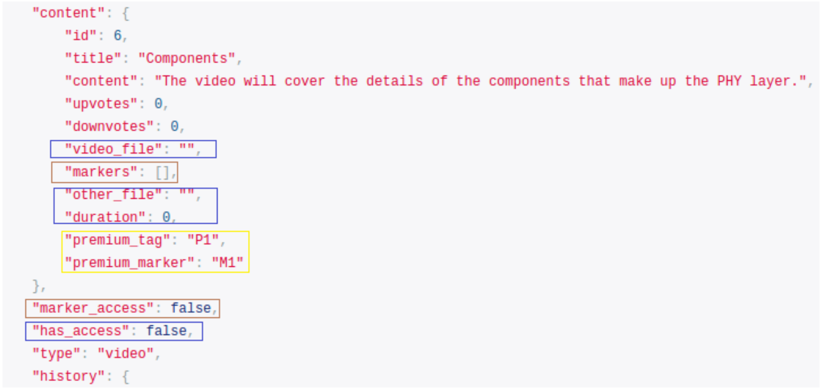
Design of student's access to content



Users access to content

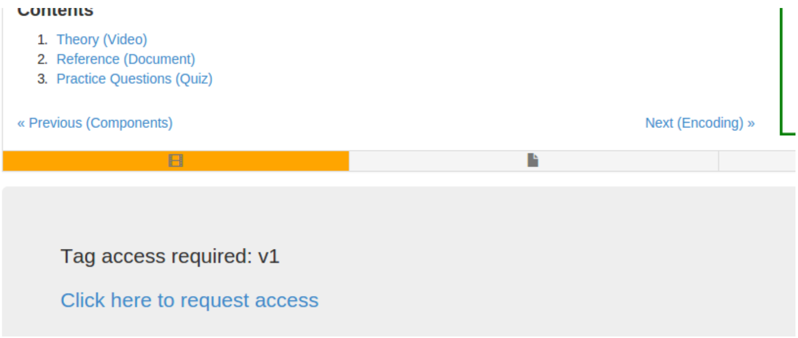
- When a user opens a concept page, the tags of elements on concept page are checked against his/her privileges
- Data fetched according to the access
- Key value pair *has_access* generated which helps in rendering the component at the client side

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": {
```

User's view of unauthorized access



Contents

1. Theory (Video)
2. Reference (Document)
3. Practice Questions (Quiz)

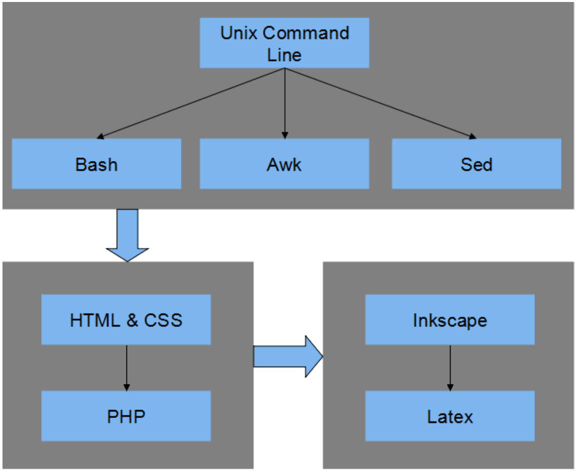
« Previous (Components)

Next (Encoding) »

Tag access required: v1

Click here to request access

Prerequisites Graph



Prerequisites Graph

Problem Statement

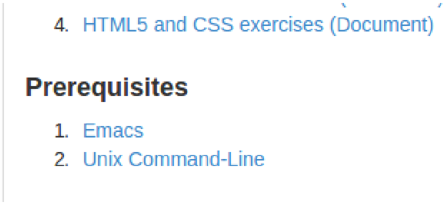
- Map the relations between concepts and display a guideline to proceed in a course
- Display link to prerequisites on the concept page for easy navigation

Specifications

- Content developer specifies the prerequisites
- Initial graph generated automatically, content developer can modify and then finalize it
- The final graph shown to the students acts as a guideline to proceed in the course

Design and Implementation

- Prerequisites stored as a list of concept id's
- Prerequisites are retrieved when the user opens a concept page, links are provided


4. HTML5 and CSS exercises (Document)

Prerequisites

1. Emacs
2. Unix Command-Line

Figure: Snap of prerequisites displayed on a concept page

Several graph generation libraries used to generate initial 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
- **GoJS:** Allowed the creation of a graph which the content developer can modify

Graphviz

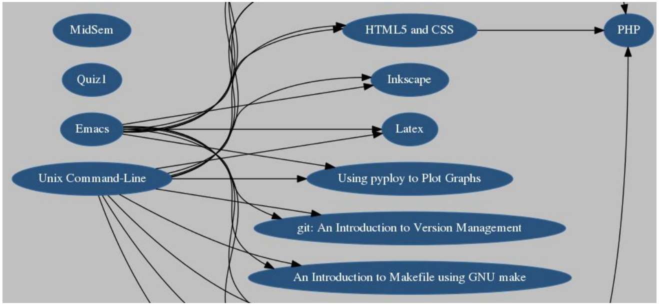

MidSem
Quiz1
Emacs
Unix Command-Line
HTML5 and CSS
PHP
Inkscape
Latex
Using pyploy to Plot Graphs
git: An Introduction to Version Management
An Introduction to Makefile using GNU make

Figure: Part of a complex graph generated using graphviz

Arbor.js

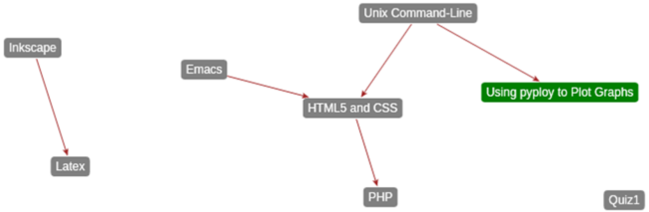


Figure: Part of a graph generated using arbor.js

Marks module

Store and display students marks

Problem Statement

- Store students marks on Bodhitree
- Enable the students to view their marks on the course page

Specifications

- The instructor can upload a CSV file containing the marks of the students
- Student can log into his Bodhitree account and can see his marks on the course page

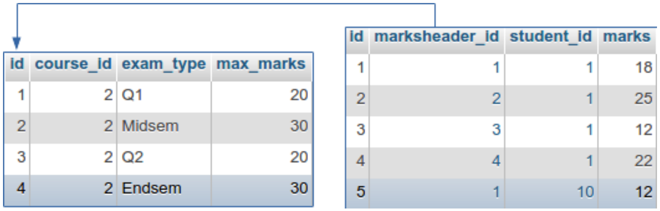
Design

- CSV file upload and storing the marks
- Displaying marks to the user
- Handling authorization

Required format of the 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
Student 3	16	24	0	22

Snapshot of the tables involved in the marks module



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

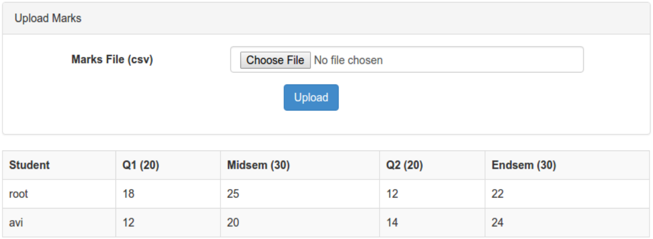
Marks Header

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

Students Marks

Figure: Database schema of the marks module

Instructor's view

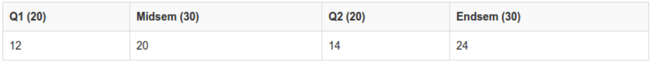

Upload Marks

Marks File (csv)

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

Student's view



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

Figure: Student 2's view of his marks

Conclusion

- Work on the marks module is done, thoroughly tested
- Access control module is implemented, initial testing done
- Some changes are suggested for the access control module
  - Changing of certain information text displayed to users
  - List of users who have premium access must collate
- Specifications for prerequisites graph module are set
  - Certain initial graphs are implemented
  - Interactivity to content developer is needed
  - Thinking about interactivity to students

Thank you!