Project Features

Tom Midson

August 4, 2014

Contents

1 Technologies

Currently the technologies being used are:

- Python
  - django for the web framework
  - pygments to highlight the students code
- Mysql to host the database needed for the application
- Possibly Python and Java for the unittests to grade work, we'll probably need to write a few scripts to batch run tests on the files in python
- HTML, CSS, JS for displaying information to the user

2 Features

1. View Reviews
2. Submit Assignment
3. Setup Assignment
4. Peer Review
5. Functionality Test
6. View results
7. Ask for help

3 Stories

3.1 Submit

1. User commits the latest version of their assignment to git/svn
2. User browsers to assignment submission on portal
3. User submits assignment

3.2 Review other students assignments

1. Browse to CPRS site
2. Choose review
3. Student highlights and annotates the student's assignment
4. Student submits comments on the file until review is complete
5. Student submits assignment

3.3 Read

1. Student opens their annotated assignments
2. Student clicks reviews recieved
3. Student can then view their code to see comments

3.4 Changing a file

1. Students can select files from the list on the side of the page

3.5 Requesting Help

1. Student is struggling with an assignment
2. Student commits their code to git/svn
3. The student submits their assignment for help, using annotations to point out problem code

3.6 Give feedback

1. Using the help system the student locates help requests
2. The student is shown the code to review
3. The student annotates the code using the same method as peer review

3.7 Reading help

1. Students can read and submit feedback to help other students
2. If the feed back solves the problem then the help request is removed

3.8 CPRS config

1. Course Coordinator browses to config page
2. Course coordinator creates or edits assignments
3. Coordinator can check and uncheck options for each assignment
4. Coordinator saves configuration

4 Models

Currently, if we use the prototype code these are the models that are defined. We'll have to analyze them to see if they'll be sufficient but they should be.

It seems like all the models use a UUID as a unique identifier.

4.1 User

Fields	Type
user_ uuid	UUIDFIeld
djangoUser	OneToOne field to a User

Methods: unicode: all this does is return the User's username.

From my research it looks like this table just uses an old style of extending django classes. Should be sufficient for our use but we'll have to see how to integrate the LTI system with the Users

4.2 SourceFolder

Fields	Type
folder_ _uuid	UUIDField
name	TextField
parent	ForeignKey(self)

Methods:

- unicode: identifies the folder
- repr: returns a representation of the folder

This and the following model are just ways to point the app at the location of the submissions

4.3 SourceFile

Fields	Type
folder	ForeignKey(SourceFolder)
file_ uuid	UUIDField
name	TextField
file	FileField

Methods:

- unicode: identifies the file
- repr: returns a representation of the file
- content: opens the file for reading and closes it when finished

See above.

4.4 SubmissionTestResults

Fields	Type
tests _completed	BooleanField

Methods:

- overall _percentage: returns the success of the users tests as a percentage
- total _test: returns the total number of tests run
- total _passes: returns the number of tests pass
- total _failures: returns the number of tests failed
- unicode: returns the number of tests passed over the number of tests

Simple way of representing the results from the unit test. Seems to be just a place holder for various functions, we could possibly refactor this.

4.5 SubmissionTest

Fields	Type
part of	ForeignKey(SubmissionTestResults)
test name	TextField
test count	PositiveIntegerField
test pass count	PositiveIntergerField

Methods:

- clean: appears the clean the results of the tests
- get _results: returns the results as a fraction
- unicode: returns the test name and a fraction of passed and total tests

Along with SubmissionTestResults this model seems to handle the running of the unit tests on the students assignments. I think I'd recommend merging with this the SubmissionTestResults unless further reading reveals that this needs to be its own model.

4.6 Assignment

Fields	Type
assignment_ uuid	UUIDField
name	TextField
repository_ format	TextField
first_ display_ date	DateTimeField
submission_ open_ date	DateTimeField
review_ open_ date	DateTimeField
review_ close_ date	DateTimeField

Methods:

- clean: appears the clean the results of the tests
- unicode: returns the test name and a fraction of passed and total tests

This is the main model for the assignments. I'd say this is the main model we'll be working with when designing the submission UI.

4.7 AssignmentSubmission

Fields	Type
submission_ uuid	UUIDField
submission_ date	DateTimeField
by	ForeignKey(User)
submission_ for	ForeignKey(Assignment)
submission_ repository	TextField
error_ occured	Boolean
root_ folder	OneToOne(SourceFolder)
test_ results	OneToOne(SubmissionTestResults)

Methods:

- clean: appears the clean the results of the tests
- unicode: returns the test name and a fraction of passed and total tests

AssignmentSubmission holds the actual submission of the student's assignments.

4.8 SourceAnnotation

Fields	Type
annotation_ uuid	UUIDField
user	ForeignKey(User)
source	ForeignKey(SourceFile)
created	DateTimeField
updated	DateTimeField
text	TextField
quote	TextField

Methods:

- str: returns a string representation of the fields

This model, along with the following two, seem to hold the information for all the code review and student help annotations. I'm unsure what SourceAnnotationTag will be used for but we might be able to merge these into a better model.

4.9 SourceAnnotationRange

Fields	Type
range_ annotaion	ForeignKey(SourceAnnotation)
start	TextField
end	TextField
startOfSet	PositiveIntegerField
endOfSet	PositiveIntegerField

4.10 SourceAnnotationTag

Fields	Type
tag_ annotaion	ForeignKey(SourceAnnotation)
tag	TextField

5 Other Code

In progress

6 Return brief

For the return brief I think we should just start working on implementing users and have a basic GUI up. I don't think it needs to really do anything just show what it'll look like. Should be too hard to put together something before next friday.